Serial Flash Mailbox Client Intel FPGA IP User Guide

Updated for Intel® Quartus® Prime Design Suite: **19.3**

Subscribe
Send Feedback

UG-20159 | 2019.09.30
Latest document on the web: PDF | HTML

Contents

Serial Flash Mailbox Client Intel FPGA IP User Guide

The Serial Flash Mailbox Client Intel FPGA IP provides access to quad serial flash devices (SPI).

For a complete list of supported flash memory devices refer to the Device Configuration - Support Center web page.

The Serial Flash Mailbox Client Intel FPGA IP core supports:

- Direct flash access (write and read) through the Avalon® Memory-Mapped (Avalon MM) interface
- Control register access for other operations through the `control and status register (CSR)` interface
- Up to 4 kilobytes (KB) data data transfers for each quad SPI read and write command
- Opcodes for the following quad SPI operations:
  - Open
  - Close
  - Set chip select
  - Read data from flash
  - Write data to flash
  - Erase sector
  - Read device register
  - Write device register
  - Send device opcode

Refer to the respective flash device datasheet for a complete list of supported operations for a particular device.

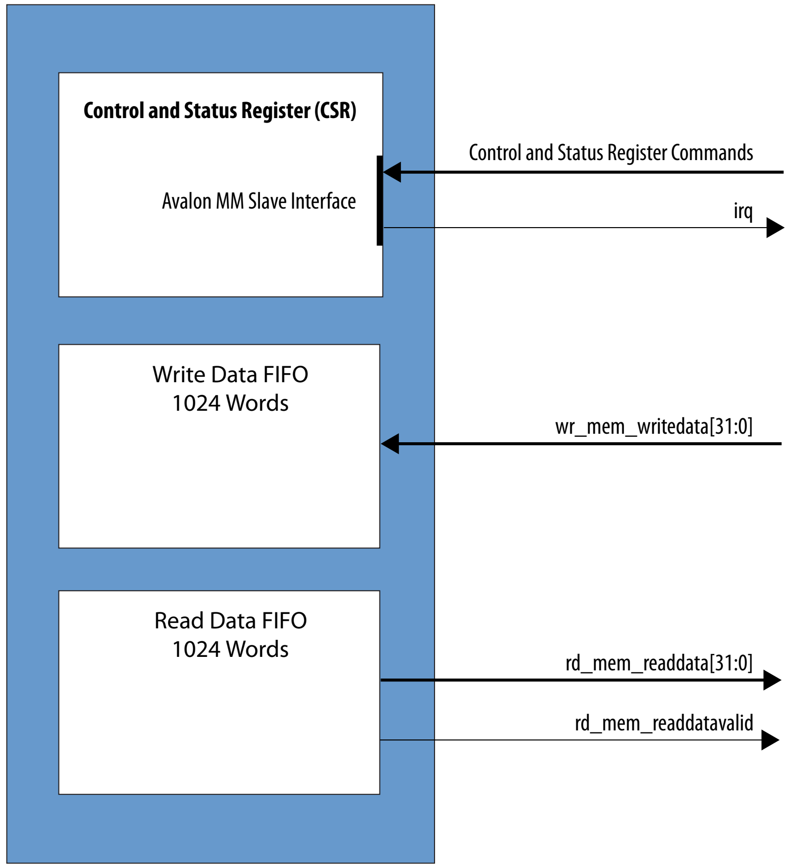
Related Information

Introduction to Intel FPGA IP Cores
: Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP cores.

Serial Flash Mailbox Client IP Modules

The following block diagram shows the modules that comprise the Serial Flash Mailbox Client IP.

Figure 1. Serial Flash Mailbox Client IP Modules



You set up quad SPI commands by writing to the CSR. The Serial Flash Mailbox Client IP sends commands to the secure device manager (SDM) in the Intel® Stratix® 10 device. The SDM controls the quad SPI device.

For write commands you prestore the write data in the write data FIFO. For read commands, you read data from the read data FIFO. The write and read data FIFOs store up to 1024 words. The write and read data FIFOs are Avalon MM slaves. The Serial Flash Mailbox Client IP asserts tine `irq` signal if the command results in an error response.

Device Family Support

The following lists the device support level definitions for Intel FPGA IPs:

- **Advance support** — The IP is available for simulation and compilation for this device family. Timing models include initial engineering estimates of delays based on early post-layout information. The timing models are subject to change as silicon testing improves the correlation between the actual silicon and the timing models. You can use this IP for system architecture and resource utilization studies, simulation, pinout, system latency assessments, basic timing assessments (pipeline budgeting), and I/O transfer strategy (data-path width, burst depth, I/O standards tradeoffs).
- **Preliminary support** — The IP is verified with preliminary timing models for this device family. The IP meets all functional requirements, but might still be undergoing timing analysis for the device family. It can be used in production designs with caution.
- **Final support** — The IP is verified with final timing models for this device family. The IP meets all functional and timing requirements for the device family and can be used in production designs.

Table 1. Device Family Support

Device Family	Support
Intel Stratix 10	Final

Note: Intel does not provide simulation modes for the Mailbox Avalon ST Client Intel FPGA IP.

Signals

Figure 2. Top-Level Signals

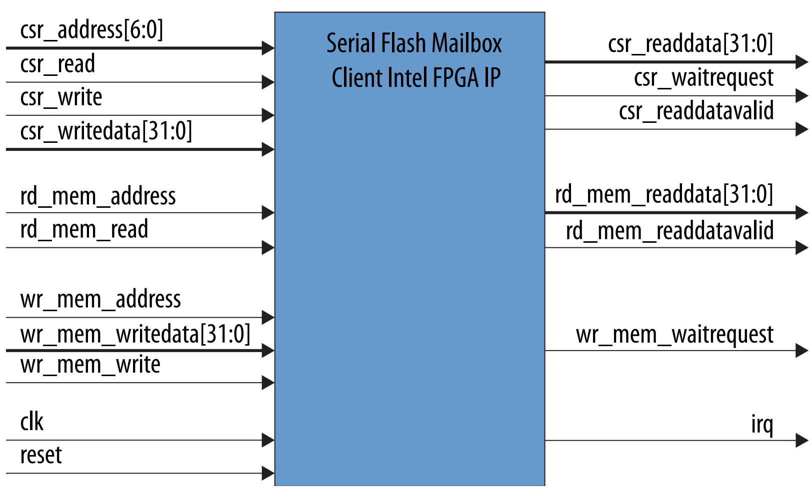


Table 2. Signal Descriptions

Signal	Width	Direction	Description
Avalon MM Control and Status Register Signals			
csr_address	7	Input	Avalon MM address bus. The address bus uses word (32-bit) addressing.
csr_read	1	Input	Avalon MM read control for the CSR.
csr_readdata	32	Output	Avalon MM read data bus from the CSR.
csr_write	1	Input	Avalon MM write control to the CSR.
csr_writedata	32	Input	Avalon MM write data bus to CSR.
csr_waitrequest	1	Output	Avalon MM waitrequest control from the CSR.
csr_readdata_valid	1	Output	Avalon MM read data valid that indicates the CSR read data is available.
Avalon MM Write Data Signals			
wr_mem_write	1	Input	Avalon MM write control to the FIFO.
wr_mem_address	1	Input	Avalon MM address.
wr_mem_writedata	32	Input	Avalon MM write data bus to the memory.
wr_mem_waitrequest	1	Output	Avalon MM waitrequest control from the memory.
Avalon MM Read Data Signals			
rd_mem_read	1	Input	Avalon MM read control to the FIFO.
rd_mem_readdata	32	Output	Avalon MM read data bus from the memory.
rd_mem_readdatavalid	1	Output	Avalon MM read data valid that indicates the memory read data is available.
rd_mem_address	1	Input	Avalon MM address.
Clock and Reset			
clk	1	Input	Input clock to clock the IP. The maximum frequency supported 250 MHz.
reset	1	Input	Synchronous reset to reset the Serial Flash Mailbox Client Intel FPGA IP. *Note:* Refer to the *AN 891: Using the Reset Release Intel FPGA IP* for more information about reset in Intel Agilex™ devices. .
irq	1	Output	The irq signal asserts if the command STATUS is not OK. Read the ISR register Rsp_status field to determine the error.

Related Information

- AN 891: Using the Reset Release Intel FPGA IP
- Avalon Interfaces Specifications
  For more information about Avalon MM interfaces, including detailed signal defintions and timing diagrams.

Register Map

Table 3. Register Map and Definitions

- Each address offset in the table represents one word of memory address space.
- All registers have a default value of 0x0 unless otherwise stated.

Offset	Name	Field Name	R/W	Width	Bit	Description
0	STATUS	Rsp_status	R	11	10:0	The status of executed commands. Refer to Response Codes on page 12 for the definitions of response codes.
1	ISR	Rddata valid	R	1	1	When 1, indicates that readdata is available in the read data FIFO. The READ_FIFO_LEVEL specifies how many words are available in the read data FIFO.
		Cmd_err	R	1	0	When 1, indicates an that the current command failed. Read the Rsp_status field of the STATUS register to determine the error condition. Assert reset to clear this bit.
2	IER	Rdat_valid_en	R/W	1	1	The enable bit for read data. The default value is 1.
		Cmd_err_en	R/W	1	0	The enable for command error responses. The default value is 1.
3	CHIP_SELECT	Chip_select	R/W	4	3:0	Write the value of the flash device you want to select.
4	OPEN	Open	W	1	0	Request exclusive access to the flash device. Write 1 to request exclusive access. The Rsp_status field of the STATUS register returns OK the SDM accepts request
5	CLOSE	Close	W	1	0	Write 1 to close exclusive access to the flash device. The Rsp_status field of the STATUS register returns OK if the SDM accepts request.
6	WR_ENABLE	Wr_enable	W	1	0	Write 1 to assert the write a enable.
7	WR_STATUS	Wr_status	W	8	7:0	Writes the STATUS register of the flash memory.
8	RD_STATUS	Rd_status	R	8	7:0	Rd_status contains the information from read STATUS register operation.(1)
9	SECTOR_ERASE	Sector_address	W	32	31:0	Erases one sector in the flash. The sector_address must be in most significant byte to least significant byte order.

continued...

(1) For Micron* devices, the flag status register provides the status of stacked devices. You can access the flag status register using the CONTROL command.

Offset	Name	Field Name	R/W	Width	Bit	Description
						For example, to erase a sector of a Micron 2 gigabit (Gb) flash at address 0x04FF0000 using the opcode method, complete the following steps: 1. Get exclusive access to the serial flash device. 2. Enable the `WR_ENABLE`: 0x00000001 3. Set `NUMB_BYTES` to 4: 0x00000004 4. Write the `CONTROL` (opcode) for the sector erase: 0xDC000021 5. Write the flash address to `WRITEDATA_0`, the lower 4 bytes of write data: 0x0000FF04 For sequential erase commands in Micron devices , read bit 7 of the`flag status` register as detailed in the flash data sheet for the device. Use the `CONTROL` register read op code to perform this read.
10	`RD_DEVICE_ID`	`Device_id`	R	32	31:0	Stores the device ID.
11 - 12	Reserved					
13	`CONTROL`	`Opcode`	R/W	8	31:24	Opcode of the flash device operation.
		Reserved[23:7]				
		`Read_data_en`	R/W	1	6	When 1, indicates the command has read data.
		`Write_data_en`	R/W	1	5	When 1, indicates the command has write data.
		Reserved[4:1]				
		`Execute`	W	1	0	Write 1 to initiate the command.
14	`NUMB_BYTES`	`Number_bytes`	R/W	4	3:0	The number of bytes to write or read from the device register (maximum 8 bytes).
15	`WRITEDATA_0`	`Writedata_0`	W	32	31:0	The lower 4 bytes of write data.
16	`WRITEDATA_1`	`Writedata_1`	W	32	31:0	The upper 4 bytes of write data.
17	`READDATA_0`	`Readdata_0`	R	32	31:0	The lower 4 bytes of read data.
18	`READDATA_1`	`Readdata_1`	R	32	31:0	The upper 4 bytes of read data.
19	Reserved					
20	`WRITE_OP`	`Write_op`	W	2	1:0	Write 2’b01 to perform write operation with address provided in offset 21 and write data in the FIFO. Write 2’b10 to flush data in write FIFO .
21	`WRITE_ADDR`	`Write_addr`	W	32	31:0	The device address for write operation.
22	`WRITE_FIFO_LEVEL`	`Wr_fifo_level`	R	32	31:0	Returns the fill level of the internal write data FIFO.
23	`READ_OP`	`Read_op`	W	2	1:0	Write 2’b01 to perform read operation with address provided in offset 24. Write 2’b10 to flush read data FIFO.

continued...

Offset	Name	Field Name	R/W	Width	Bit	Description
24	READ_ADDR	Read_addr	R/W	32	31:0	The device address for read operation.
25	READ_WORDS	Read_words	R/W	32	31:0	Number of words to read from device (maximum is 4 KB.)
26	READ_FIFO_LEVEL	Read_fifo_level	R	32	31:0	Specifies the fill level of the internal read data FIFO.

Related Information

Response Codes on page 12

Operation Commands

Table 4. Command List and Description

Command	Code (Hex)	Command Length [(2)]	Response Length [(2)]	Description
QSPI_OPEN	32	0	1	Requests exclusive access to the quad SPI. The SDM accepts the request if the quad SPI is not in use and the SDM is not configuring the device. Returns OK if the SDM grants access. Returns the ALT_SDM_MBOX_RESP_DEVICE_BUSY when the quad SPI flash is busy. *Note:* The SDM grants exclusive access to the client using this mailbox. Other clients cannot access the quad SPI until the active client relinquishes access using the QSPI_CLOSE command.
QSPI_CLOSE	33	0	1	Closes the exclusive access to the quad SPI interface.
QSPI_SET_CS	34	1	1	Specifies one of the attached quad SPI devices via the chip select lines. Takes a one-word argument as described below: • Bits[31:28]: Flash device to select. The value 4'b0000 selects the flash that corresponds to nCSO[0]. nCSO[0] is the only signal that the FPGA can use to access the quad SPI flash device. The HPS can use nCSO[3:1] to access HPS data. • Bits[27:0]: Reserved (write as 0). The HPS can use nCSO[3:1] to access 3 additional quad SPI devices. This command is optional for the AS x4 configuration scheme. Is required for all other configuration schemes.
QSPI_READ	3A	2	N	Reads the attached quad SPI device. The maximum read size is 4 kilobytes (KB). Takes two arguments: • The quad SPI flash address (one word). The address must be word aligned. The device returns the 0x1 error code for non-aligned addresses. • Number of words to read (one word). When successful returns OK followed by the read data from the quad SPI device. A failure response returns an error code. For a partially successful read, QSPI_READ may erroneously return the OK status. *Note:* You cannot run the QSPI_READ command while device configuration is in progress.
QSPI_WRITE	39	2+N	0	Writes data to the quad SPI device. Takes three arguments:

continued...

(2) This number does not include the command and response header.

Command	Code (Hex)	Command Length [(2)]	Response Length [(2)]	Description
				• The flash address offset (one word). The write address must be word aligned. The device returns error code 0x3FF for non-aligned addresses. • The number of words to write (one word). • The data to be written (one or more words). A successful write returns the OK response code. To prepare memory for writes, Intel recommends using the QSPI_ERASE command before issuing this command. *Note:* You cannot run the QSPI_WRITE command while device configuration is in progress.
QSPI_ERASE	38	2	0	Erases a sector of the quad SPI device. Takes two arguments: • The flash address offset to start the erase (one word). The address must be the start address of a sector within the flash memory; consequently, the address must be 64 KB aligned. Returns an error for non-64 KB aligned addresses. • The number of words to erase specified in multiples of 0x4000 words. A successful erase returns the OK response code.
QSPI_READ_DEVICE_REG	35	2	N	Reads registers from the quad SPI device. The maximum read is 8 bytes. Takes two arguments. • The opcode for the read command. • The number of bytes to read. A successful read returns the OK response code followed by the data read from the device. Pads data that is not a multiple of 4 bytes to the next word boundary.
QSPI_WRITE_DEVICE_REG	36	2+N	0	Writes to registers of the quad SPI. The maximum write is 8 bytes. Takes three arguments: • The opcode for the write command. • The number of bytes to write. • The data to write. To perform a sector erase or sub-sector erase, you must specify the serial flash address in most significant byte (MSB) to least significant byte (LSB) order as the following example illustrates. To erase a sector of a Micron 2 gigabit (Gb) flash at address 0x04FF0000 using the QSPI_WRITE_DEVICE_REG command, write the flash address in MSB to LSB order as shown here: Header: 0x00003036 Opcode: 0x000000DC Number of bytes to write: 0x00000004 Flash address: 0x0000FF04 A successful write returns the OK response code. This command pads data that is not a multiple of 4 bytes to the next word boundary. To perform a sector erase or sub-sector erase, you must specify the serial flash address in most significant byte (MSB) to least significant byte (LSB) order as the following example illustrates. To erase a sector of Micron's 2 gigabit (Gb) at flash address 0x04FF0000 using the QSPI_WRITE_DEVICE_REG command, write the flash address in MSB to LSB order as shown: Header: 0x00003036 Opcode: 0x000000DC Number of bytes to write: 0x00000004

continued...

(2) This number does not include the command and response header.

Command	Code (Hex)	Command Length [(2)]	Response Length [(2)]	Description
				Flash address: 0x0000FF04
QSPI_SEND_DEVICE_OP	37	1	0	Sends a command opcode to the quad SPI. Takes one argument: • The opcode to send the quad SPI device. A successful command returns the OK response code.

Table 5. CONFIG_STATUS and RSU_STATUS Major Error Code Descriptions

Major Error Code	Error Type	Description
0xF001	BITSTREAM_ERROR	Potential unsigned bitstream used. Ensure the bitstream is signed with the correct key.
0xF002	HARDWARE_ACCESS_FAILURE	Failure to communicate to PMBus-compliant voltage regulator. Check your power management and smart voltage identification (SmartVID) parameter settings and PMBus interface connections.
0xF003	BITSTREAM_CORRUPTION	Bitstream is corrupt. Ensure the bit stream in configuration device or flash is not corrupt.
0xF004	INTERNAL_ERROR	This major code may indicate the following error events: • An error in the SDM Crypto IP task. • An RSU operation error. Refer to Table 6 on page 11 for more information.
0xF005	DEVICE_ERROR	Indicates an SDM internal device error. The following errors are possible: • A device cleaning failure • An HPS configuration failure Contact your local Field Applications Engineer (FAE). Alternatively, submit a Service Request on the My Intel support page to get the support on capturing the error log for further debug.
0xF006	HPS_WATCHDOG_TIMEOUT	HPS watchdog timeout failure. Ensure that your design resets the watchdog timer correctly.
0xF007	INTERNAL_UNKNOWN_ERROR	Indicates an internal device error due to an unknown task. You can contact your local Field Applications Engineer (FAE). Alternatively, submit a Service Request on the My Intel support page to get the support on capturing the error log for further debug.

Table 6. CONFIG_STATUS and RSU_STATUS Minor Error Code Descriptions

Minor Error Code	Error Type	Description
0xD001	RSU_CMF_AUTH_ERR	Authentication failure for the firmware.
0xD002	RSU_USER_AUTH_ERR	Authentication failure for the design.
0xD003	RSU_CMF_DESC_SHA_MISMATCH	The SHA does not match for the firmware descriptor.
0xD004	RSU_POINTERS_NOT_FOUND_ERR	Unable to read data from boot ROM on first boot after the device exits power-on reset (POR).

continued...

(2) This number does not include the command and response header.

Minor Error Code	Error Type	Description
0xD005	RSU_QSPI_REQ_CHANGE	Unable to configure the quad SPI flash during RSU initialization.
0xD006	RSU_FACTORY_IMAGE_FAILED	Failed to load any image, including the factory image.
0xD007	RSU_CMF_TYPE_ERR	The firmware version does not match the version that was previously loaded.

Response Codes

The response code for each command along with read data, if applicable. Read the Rsp_status field of the STATUS register bit to determine if the command completed without errors.

Table 7. Error Codes

Value (Hex)	Error Code Response	Description
0	OK	Indicates that the command completed successfully. A command may erroneously return the OK status if a command, is partially successful. For example, if a read operation fails after returning some data.
1	INVALID_COMMAND	Indicates that the command is incorrectly formatted.
2	UNKNOWN_BR	Indicates that the command code is not understood.
3	UNKNOWN	Indicates that the currently loaded firmware cannot decode the command code.
4	INVALID_COMMAND_PARAMETERS	The length or indirect setting in header is not valid. Or the command data is invalid.
5	COMMAND_INVALID_ON_SOURCE	Command is from a source for which it is not enabled.
6	CLIENT_ID_NO_MATCH	Indicates that the Client ID requesting quad SPI or SD MMC access does not have exclusive access.
7	INVALID_ADDRESS	The address is invalid. This error indicates one of the following conditions: • An unaligned address • An address range problem • A read permission problem
8	TIMEOUT	The command timed out.
9	HW_NOT_READY	The hardware is not ready. Can indicate either an initialization or configuration problem.
100	NOT_CONFIGURED	Indicates that the device is not configured.
1FF	ALT_SDM_MBOX_RESP_DEVICE_BUSY	Indicates that the device is busy.
2FF	ALT_SDM_MBOX_RESP_NO_VALID_RESP_AVAILABLE	Indicates that there is no valid response available.
3FF	ALT_SDM_MBOX_RESP_ERROR	General Error.

Related Information

Register Map on page 7

Using the Serial Flash Mailbox Client Intel FPGA IP

The following topics lists the steps you must follow for CSR write and read operations.All interfaces are Avalon MM compliant. Refer to the *Avalon Interface Specification* for more information Avalon interfaces.

Related Information

Introduction to Avalon Memory-Mapped Interfaces
Includes detailed signal definitions and timing diagrams.

Control and Status register (CSR) Operation

Follow these steps to perform a read or write to a specific address offset using the Serial Flash Mailbox Client Intel FPGA IP CSR.

1. Assert the `csr_write` or `csr_read` signals while the `csr_waitrequest` signal is low. If the `csr_waitrequest` signal is high, the `csr_write` or `csr_read` signals must be kept high until the `csr_waitrequest` signal goes low.
2. Depending on the operation, perform the following steps:
   — For read operations, set the address value on the `csr_address` bus.
   — For write operations, set the address value on the `csr_address` bus and the value data on the `csr_writedata` bus.
3. For read operations, you can retrieve the data after the `csr_readdatavalid` signal is high.

Write Operation

Complete the following steps to perform a write operation: The maximum write size is 4 KB (1000 words).

1. Request exclusive access to the flash memory using the `QSPI_OPEN` command.
2. Select the flash device using the `QSPI_SET_CS` command.
3. Erase the flash device using the `QSPI_ERASE` command.
4. Flush the write data FIFO by writing `2b'10` using the `WRITE_OP` command.
5. Pre-store the data you want to write to the flash device in the write data FIFO via write data interface. Write the write data FIFO:

   Note: The interface backpressures when the write data FIFO is full.

   a. Assert the `wr_mem_write` signal while the `wr_mem_waitrequest` signal is low. If the `wr_mem_waitrequest` signal is high, the `wr_mem_write` signal must remain high until the `wr_mem_waitrequest` signal goes low.)
   b. Write the address value to the `wr_mem_address` bus. Write the data value to `wr_mem_writedata` bus.

      Note: Refer to the base address assigned to `wr_mem` bus for Serial Flash Mailbox Client Intel FPGA IP in the Intel Quartus® Prime Platform Designer for list of address values that you can write.

   c. Repeat step *a* and *b* to continuously pre-store the data into the write data FIFO. The FIFO has 1024 words. Consequently, the maximum write is 1024 words.

d. De-assert the wr_mem_write signal after writing all of the data into the write data FIFO.

e. Optional: You can read the fill level of the internal write data FIFO using the WRITE_FIFO_LEVEL command determine if the write data FIFO is full.

6. Read the Rsp_status field of the STATUS register to check the status of the write.

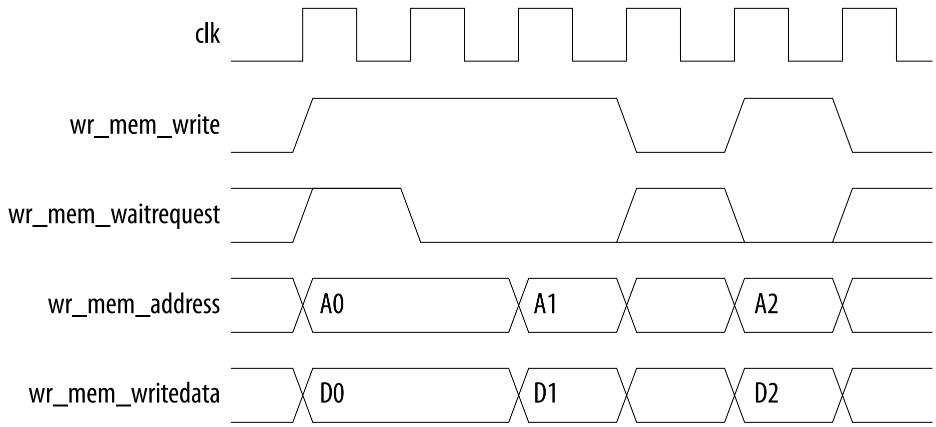
7. Start the write operation by transferring the data from the write data FIFO into the flash device by writing 2'b01 to the WRITE_OP command.

8. Poll the Cmd_err field of the ISR register to check the status of the write transaction. The ISR writes a value of one to the Cmd_err field of the ISR register if a write is unsuccessful. You can also check the Rsp_status field of the STATUS register.

9. Repeat step 3 on page 13 to 8 on page 14 to continue to perform the subsequent write operation.

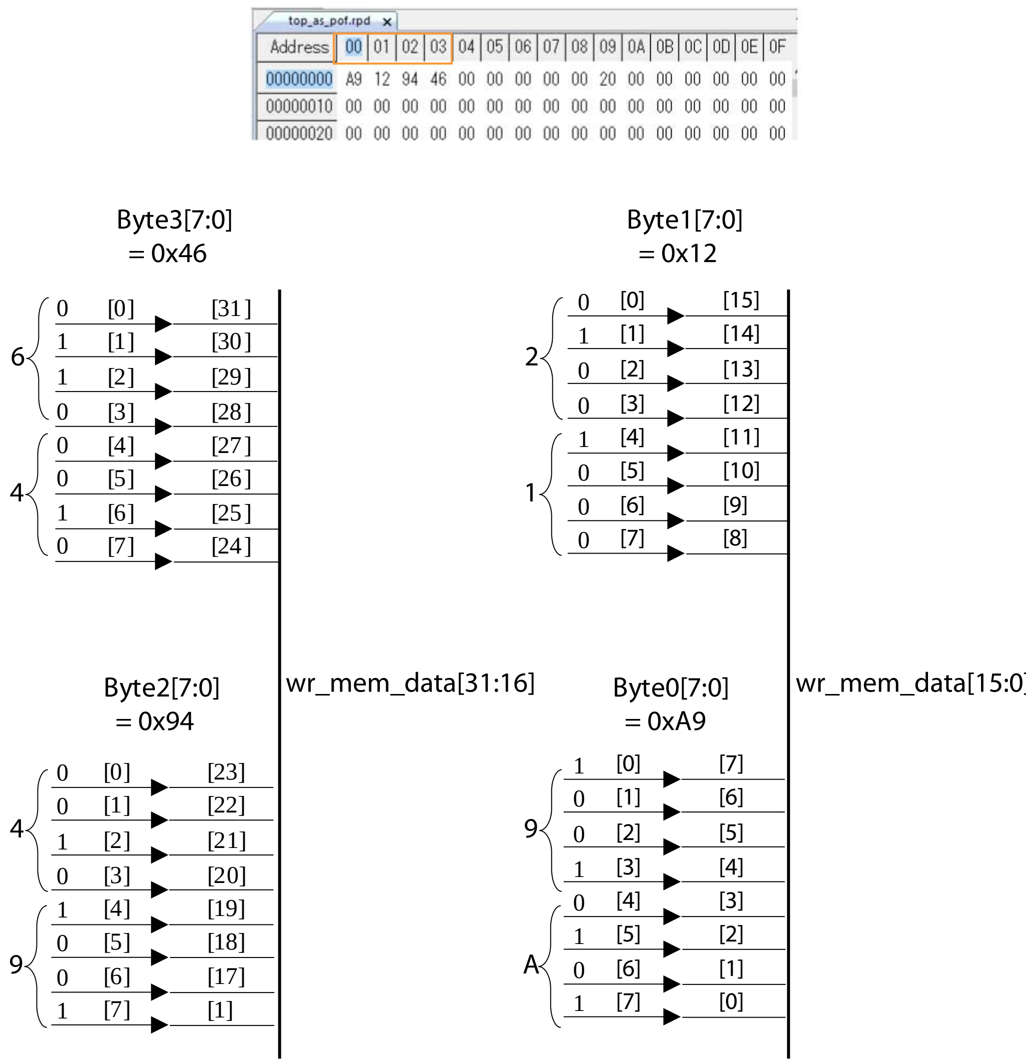
10. Release exclusive access to the flash device using the CLOSE command.

Figure 3. Write Operation Example Timing Diagram



You can use the Serial Flash Mailbox Client Intel FPGA IP to write the raw programming data (.rpd) file into the flash device. By default, the .rpd file is little-endian. To read flash data back correctly, you must transmit the data to flash memory in big-endian format. First reverse the bit order for each byte. Then write the data to flash from the least significant bit (LSB) to most significant bit (MSB) order. The following figure illustrates this process.

Figure 4. Connections for Little Endian Format When Bit Swap Is Off



Note:

Related Information

Generating Programming Files using Convert Programming Files in the Intel Stratix 10 Configuration User Guide

Read Operation

Complete the following steps to perform a read operation.The maximum read size is 4 KB (1000 words).

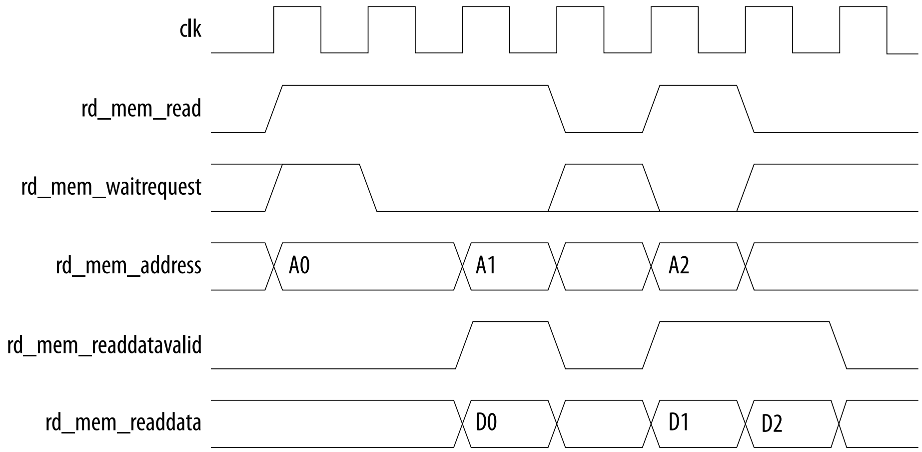
1. Request exclusive access to the flash memory using the `OPEN` command.
2. Select the flash device using the `CHIP_SELECT` command.
3. Specify the flash device address using the `READ_ADDR` command.

4. Specify the number using the READ_WORDS command.
5. Flush the read data FIFO before performing any read operations by writing 2b’10 to the READ_OP command.
6. Start the read operation by transferring data from flash to the read data FIFO by writing 2’b01 to the READ_OP command.
7. Poll the Rddata valid of the ISR register to when the data stored in read data FIFO is ready to read. You can also read the fill level of the internal read data FIFO using the READ_FIFO_LEVEL command. You can also read the Cmd_err field of the ISR register to check the status of the read transaction. The ISR writes a value of one to the Cmd_err field of the ISR register if a read is unsuccessful. You can also check the Rsp_status field of the STATUS register.
8. Read the data stored in read data FIFO via read data interfaces.
   a. Assert the rd_mem_read signal while the rd_mem_waitrequest signal is low. If the rd_mem_waitrequest signal is high, the rd_mem_read signal must be kept high until the wr_mem_waitrequest signal goes low.
   b. Set the address value at the rd_mem_address bus.

      Note: Refer to the base address assigned to the rd_mem bus for the Serial Flash Mailbox Client Intel FPGA IP in the Intel Quartus Prime Platform Designer for the assigned addresses. .
   c. Read the rd_mem_readdata bus if the rd_mem_readdatavalid signal is asserted.
   d. Repeat steps *a* to *c* to continuously read the data from the read data FIFO.
   e. De-assert the rd_mem_read signal once you have completed reading the data from the read data FIFO.
   f. Optional: You can read the fill level of the internal read data FIFO using the READ_FIFO_LEVEL command.
9. Repeat step 3 on page 15 to 8 on page 16 to continue to performing read operations.

   Note: You can check the STATUS register each time you send a command to ensure that the command completed successfully.

Figure 5. Read Operation Example Timing Diagram



10. Release exclusive access to the flash device using the `CLOSE` command.

Design Example

The Serial Flash Mailbox Client Intel FPGA IP example design is available in the Intel Design Store. includes the following functions:

- Creates a flash image containing a configuration bitstream. This image includes the Serial Flash Mailbox Client Intel FPGA IP.
- Programs the flash memory using the configuration bitstream.
- Reads the status register of the flash memory device.
- Reads the flash memory device ID.
- Reads the flash memory device ID using the `CONTROL` command.
- Erases flash memory.
- Reads flash memory.
- Writes flash memory.

Related Information

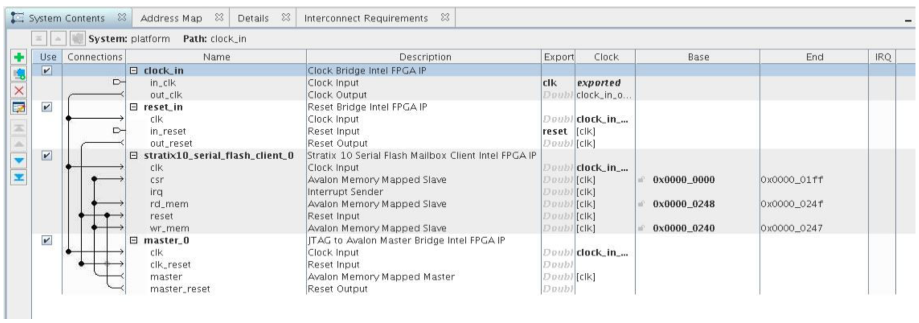
Intel Stratix 10 Serial Flash Mailbox Client Intel FPGA IP Core Design Example

Prerequisites

You can create a very simple Intel Quartus Prime Pro Edition Platform Designer design example to exercise the Serial Flash Mailbox Client Intel FPGA IP. This design example must meet the following hardware and software requirements:

- You should be running the Intel Quartus Prime Pro Edition Edition software version 18.0 or later.
- Your Platform Designer design example should include the components in the following figure:

Figure 6. Required Communication and Host Components for the Serial Flash Mailbox Client Intel FPGA IP Design Example



- — Instantiate the JTAG to Avalon Master as host.
- — Instantiate the Serial Flash Mailbox Client Intel FPGA IP.
- — Connect the Serial Flash Mailbox Client Intel FPGA IP to JTAG to Avalon Master Bridge.
- — Set base addresses for `csr`, `rd_mem`, and `wr_mem`.

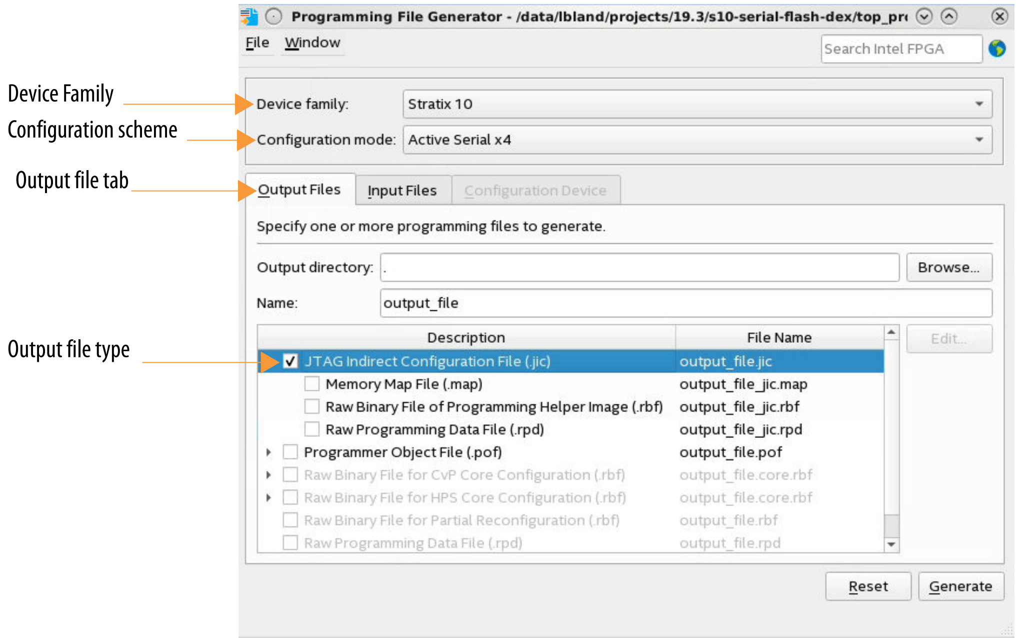
- You should be using the Intel Stratix 10 SoC Development Kit.

Generating the Configuration Bitstream

Use the Programming File Generator to generate configuration bitstream.

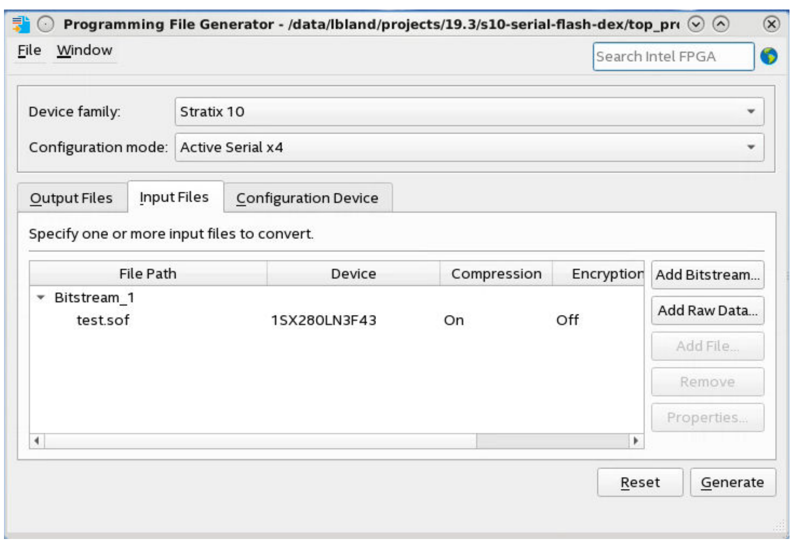
1. On the Intel Quartus Prime File menu select **Programming File Generator**.
2. For **Device family** specify **Intel Stratix 10**.
3. For **Configuration mode** specify **Active Serial x4**
4. On the **Output Files** tab, select the **JTAG Indirect Configuration File (.jic)** output file type.

Figure 7. Output Files Tab

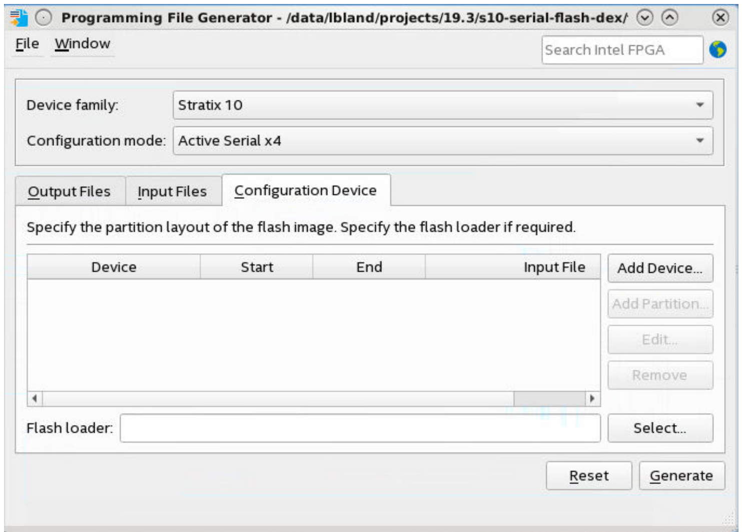


5. On the **Input Files** tab, click **Add Bitstream** and browse to your `.sof`.

Figure 8. Input Files Tab



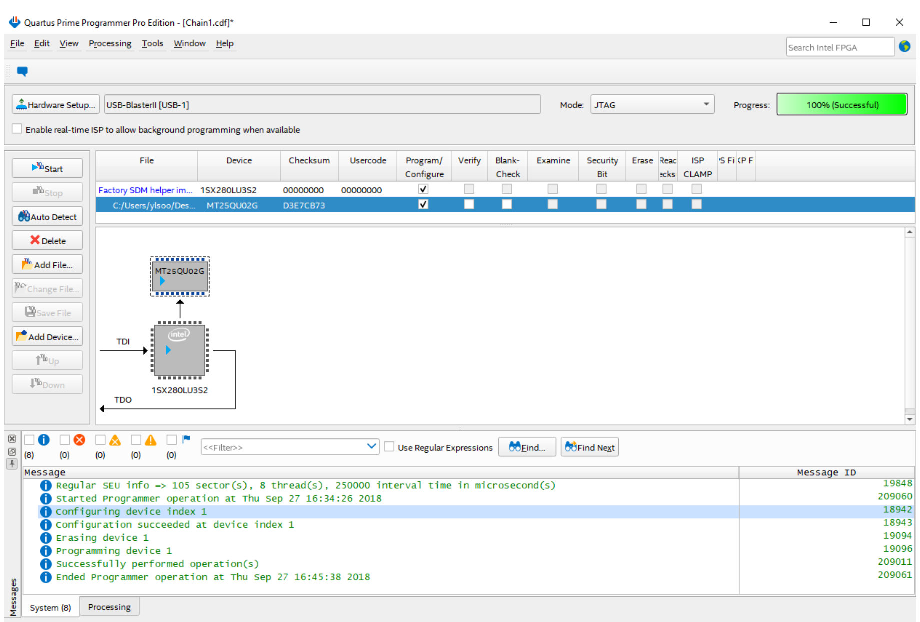
6. On the **Configuration Device** tab select **Add Device** and select your device from the list of flash devices.



7. Click **Generate** to generate the configuration bitstream.

Programming the Flash Memory with the Configuration Bitstream

1. Open **Programmer** and click **Add File**. Select the generated JIC programming file (`.jic`) and click **Open**.
2. Check the **Program/Configure** check box for the attached `.jic` file.
3. To begin programming the flash memory with the configuration bitstream, click **Start**.
4. Configuration is complete when the progress bar reaches 100%. Power cycle the board. The Intel Stratix 10 device configures automatically using the Active Serial (AS) configuration scheme.



Reading the Flash Memory Device Status Register

Here are the steps to read the flash memory device status register.

```
# Set base address according to Platform Designer system
set CSR 0x00000000
set WR_MEM 0x00000240
set RD_MEM 0x00000248
# Set the variables to their respective offset
set AsmiCmdStatus          [expr $CSR + [expr 0x0  << 2]]
set AsmiIsr                    [expr $CSR + [expr 0x1  << 2]]
set AsmiIer                   [expr $CSR + [expr 0x2  << 2]]
set AsmiChipSelect             [expr $CSR + [expr 0x3  << 2]]
set AsmiOpen               [expr $CSR + [expr 0x4  << 2]]
set AsmiClose               [expr $CSR + [expr 0x5  << 2]]
set AsmiWrEnable               [expr $CSR + [expr 0x6  << 2]]
set AsmiWrStatus               [expr $CSR + [expr 0x7  << 2]]
set AsmiRdStatus              [expr $CSR + [expr 0x8  << 2]]
set AsmiSectorErase        [expr $CSR + [expr 0x9  << 2]]
set AsmiDeviceId              [expr $CSR + [expr 0xa  << 2]]
set AsmiControl              [expr $CSR + [expr 0xd  << 2]]
set AsmiNumbByte          [expr $CSR + [expr 0xe  << 2]]
set AsmiWriteData0         [expr $CSR + [expr 0xf  << 2]]
set AsmiWriteData1         [expr $CSR + [expr 0x10  << 2]]
set AsmiReadData0          [expr $CSR + [expr 0x11  << 2]]
set AsmiReadData1          [expr $CSR + [expr 0x12  << 2]]
set AsmiWriteOp              [expr $CSR + [expr 0x14  << 2]]
set AsmiWriteAddr              [expr $CSR + [expr 0x15  << 2]]
set AsmiWriteFifoLevel        [expr $CSR + [expr 0x16  << 2]]
set AsmiReadOp              [expr $CSR + [expr 0x17  << 2]]
set AsmiReadAddr              [expr $CSR + [expr 0x18  << 2]]
set AsmiReadNumbWords        [expr $CSR + [expr 0x19  << 2]]
set AsmiReadFifoLevel        [expr $CSR + [expr 0x1A  << 2]]

# Assign variable "m" to the string that is the 0th element in the list
returned by get_service_paths master
set m [ lindex [ get_service_paths master ] 0 ]

# Open the connection to the master module
open_service master $m
# Assign variable "m" to the string that is the 0th element in the list
returned by get_service_paths master
set m [ lindex [ get_service_paths master ] 0 ]

# Open the connection to the master module
open_service master $m


# Write Offset 4 to request access to the flash memory device.
master_write_32 $m $AsmiOpen 0x1

# Write Offset 3 to select the 1st flash memory device attached to the IP.
master_write_32 $m $AsmiChipSelect 0x0

# Read Offset 8 for status register of the device.
master_read_32 $m $AsmiRdStatus 1

# Write Offset 5 to close access to the flash memory device.
master_write_32 $m $AsmiClose 0x1
```

Reading the Flash Memory Device ID

Here are the steps to read the Device ID from a flash memory device:

```
# Write Offset 4 to request access to the flash memory device.
master_write_32 $m $AsmiOpen 0x1
```

```
# Write Offset 3 to select the 1st flash memory device attached to the IP.
master_write_32 $m $AsmiChipSelect 0x0

# Read Offset 10 to obtain the device ID.
master_read_32 $m $AsmiDeviceId 1

# Write Offset 5 to close access to the flash memory device.
master_write_32 $m $AsmiClose 0x1
```

Reading the Flash Memory Device ID Using the Control Command

Here are the steps to to read the flash memory device ID using the CONTROL command:

```
# Write Offset 4 to request access to the flash memory device.
master_write_32 $m $AsmiOpen 0x1

# Write Offset 3 to select the 1st flash memory device attached to the IP.
master_write_32 $m $AsmiChipSelect 0x0

# Writing the command argument to Offset 14 (specify the number bytes to 0x4)
master_write_32 $m $AsmiNumbByte 0x4

# Writing the command argument to Offset 13 (the opcode to read device ID is
0xAF000041)
master_write_32 $m $AsmiControl 0xAF000041

# Read Offset 17 to determine the lower 4 bytes of read data where the device
ID obtained from control command is stored at.
master_read_32 $m $AsmiReadData0 1
# Write Offset 5 to close access to the flash memory device.
master_write_32 $m $AsmiClose 0x1
```

Erasing Flash Memory

Here are the steps to erase flash memory:

```
# Write Offset 4 to request access to the flash memory device.
master_write_32 $m $AsmiOpen 0x1

# Write Offset 3 to select the 1st flash memory device attached to the IP.
master_write_32 $m $AsmiChipSelect 0x0

# Write Offset 6 to perform write enable operation to the device.
master_write_32 $m $AsmiWrEnable 0x1

# Writing the address argument to Offset 9 (Perform Sector Erase on address
0x03FF0000)
master_write_32 $m $AsmiSectorErase 0x03FF0000

# Write Offset 5 to close access to the flash memory device.
master_write_32 $m $AsmiClose 0x1
```

Reading Flash Memory

Here are the steps to read flash memory:

```
# Write Offset 4 to request access to the flash memory device.
master_write_32 $m $AsmiOpen 0x1

# Write Offset 3 to select the 1st flash memory device attached to the IP.
master_write_32 $m $AsmiChipSelect 0x0
```

```
# Writing the address argument to Offset 24 (specify the device address for
read operation to 0x03FF0000)
master_write_32 $m $AsmiReadAddr 0x03FF0000

# Writing the command argument to Offset 25 (specify the number of words to
read from device is 1)
master_write_32 $m $AsmiReadNumbWords 0x1

# Writing the command argument to Offset 23 (Specify 0x2 to flush out data
inside read FIFO)
master_write_32 $m $AsmiReadOp 0x2

# Writing the command argument to Offset 23 (Specify 0x1 to perform read
operation)
master_write_32 $m $AsmiReadOp 0x1


# Read Offset 26 to determine the fill level of the internal read data FIFO.
master_read_32 $m $AsmiReadFifoLevel 1

# Read the data stored in read data FIFO via the base address of rd_mem in the
IP.
master_read_32 $m $RD_MEM 1

# Write Offset 5 to close access to the flash memory device.
master_write_32 $m $AsmiClose 0x1
```

Writing Flash Memory

Here are the steps to write to flash memory:

```
# Write Offset 4 to request access to the flash memory device.
master_write_32 $m $AsmiOpen 0x1

# Write Offset 3 to select the 1st flash memory device attached to the IP.
master_write_32 $m $AsmiChipSelect 0x0

# Write Offset 6 to perform write enable operation to the device.
master_write_32 $m $AsmiWrEnable 0x1

# Writing the command argument to Offset 20 (Specify 0x2 to flush out data
inside write FIFO)
master_write_32 $m $AsmiWriteOp 0x2

# Pre-store the data that you want to write into flash memory in write data
FIFO via the base address of wr_mem in the IP (Specify 0x11223344 to write into
write data FIFO)
master_write_32 $m $WR_MEM 0x11223344

# Read Offset 22 to determine the fill level of the internal write data FIFO.
master_read_32 $m $AsmiWriteFifoLevel 1

# Writing the address argument to Offset 21 (specify the device address for
write operation to 0x03FF0000)
master_write_32 $m $AsmiWriteAddr 0x03FF0000

# Writing the command argument to Offset 20 (Specify 0x1 to perform write
operation)
master_write_32 $m $AsmiWriteOp 0x1

# Write Offset 5 to close access to the flash memory device.
master_write_32 $m $AsmiClose 0x1
```

Serial Flash Mailbox Client Intel FPGA IP Core User Guide Archives

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
19.1	Serial Flash Mailbox Client Intel FPGA IP
18.0	Serial Flash Mailbox Client Intel FPGA IP

Document Revision History for the Serial Flash Mailbox Client Intel FPGA IP User Guide

Document Version	Intel Quartus Prime Version	Changes
2019.09.30	19.3	Made the following changes: • Changed the name of this IP from to Serial Flash Mailbox Client Intel FPGA IP. • Added support for Micron and Macronix flash devices. • Added *Operation Commands* topic covering the quad SPI commands available to access flash memory devices. • Added the missing`irq` signal description. • Added *Serial Flash Mailbox Client* topic. • Made the following changes to the *Write Operation* and *Read Operation* steps: 1. The operations include the `QSPI_OPEN` and `QSPI_SET_CS` commands before specifying the flash address. 2. The operations include the `QSPI_CLOSE` command after completing the write or read. • Corrected bit ordering for the `SECTOR_ERASE` and `RD_DEVICE_ID` commands. The correct bit ordering is [31:0]. • Made the sector erase command mandatory for write operations. • Removed references to electrically programmable configuration quad-serial low voltage (EPCQ-L) devices. As of 2018, EPCQ-L are obsolete. • Added an example to illustrate bit swapping for the `.rpd` format which is little endian. • Edited the entire user guide for clarity and style. • Corrected minor errors and typos.
2019.05.17	19.1	• Updated Table: *Signal Description* to add a note regarding IP core instantiation guidelines to the `reset` signal. • Added the *Archives* topic.
2019.04.01	19.1	Updated Table: *Register Map and Definitions* to update the field name for ISR from Cmd_err1 to Cmd_err.
2018.12.24	18.0	• Added a design example section. • Updated Table: *Register Map and Definitions*. • Updated Table: *Response Codes*. • Corrected minor typographical errors.
2018.05.07	18.0	Initial release.