ASMI SPI Flash Controller Core User Guide

UG-CORE-ASMIFLASH-v3.9
June 2023
www.efinixinc.com

Contents

Introduction

The ASMI SPI flash controller core implements a basic active serial memory interface (ASMI) and supports flash memory devices. This easy-to-use core lets you control the flash device on your board without having to worry about the details of the serial interface or the flash device's read and write protocols.

Use the IP Manager to select IP, customize it, and generate files. The ASMI SPI flash controller core has an interactive wizard to help you set parameters. The wizard also has options to create a testbench and/or example design targeting an Efinix® development board.

Features

- Simple active serial memory interface (ASMI)
- Supports page write, page write quad, fast read, fast read dual, fast read quad, block erase and sector erase functions
- Supports release from power-down state
- Verilog RTL and simulation testbench
- Includes example designs targeting:
    — Trion® T120 BGA324 Development Board
    — Trion® T20 BGA256 Development Board
    — Titanium Ti60 F225 Development Board
    — Titanium Ti180 M484 Development Board

Device Support

Table 1: ASMI SPI flash controller Core Device Support

FPGA Family	Supported Device
Trion	All
Titanium	All

Resource Utilization and Performance

> **Note:** The resources and performance values provided are based on some of the supported FPGAs. These values are just guidance and change depending on the device resource utilization, design congestion, and user design.

Table 2: Titanium Resource Utilization and Performance

FPGA	Logic Elements (Logic, adders, Flipflops, etc.)	Memory Blocks	DSP Blocks	f_{MAX} (MHz)	Efinity® Version[1]
Ti60 F225 C4	1,626/60,800 (2.67%)	1/256 (0.39%)	0	443	2022.2
Ti180 M484 C4	1,626/172,800 (0.94%)	1/1,280 (0.08%)	0	414	2022

Table 3: Trion Resource Utilization and Performance

FPGA	Logic Utilization (LUTs)	Registers	Memory Blocks	Multipliers	f_{MAX} (MHz)	Efinity® Version[1]
T20 BGA256 C4	1,597	305	2	0	131	2022.2
T120 BGA324 C4	1,597	305	2	0	129	2022.2

Release Notes

You can refer to the IP Core Release Notes for more information about the IP core changes. The IP Core Release Notes is available in the **Efinity Downloads** page under each Efinity software release version.
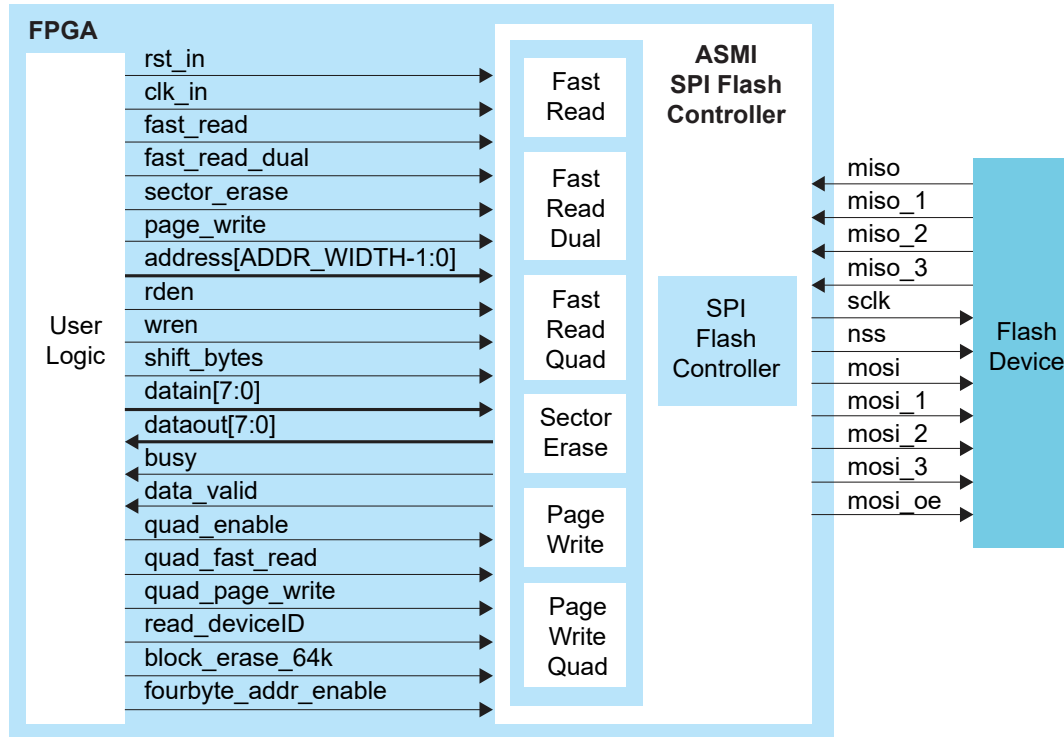
> **Note:** You must be logged in to the Support Portal to view the IP Core Release Notes.

---

[1] Using Verilog HDL.

Functional Description

The following figure shows the Trion® FPGA interface with flash device's general-purpose memory using ASMI SPI flash controller core.

Figure 1: ASMI SPI Flash Controller System Block Diagram

ASMI SPI Flash Controller Core Ports

Table 4: User Interface Ports

Port	Direction	Description
clk_in	Input	System clock.
rst_in	Input	Active high synchronous reset. During the reset, the controller sends a software reset to the flash device.
fast_read	Input	Active-high port that executes fast read operation. When asserted, the core performs a fast read operation from the flash address specified by address[n:0].
fast_read_dual	Input	Active-high port that executes a fast read dual operation.
sector_erase	Input	Active-high port that executes a sector erase operation. When asserted, the core performs a sector erase operation on the flash address specified by address[n:0].
page_write	Input	Active-high port that executes a write operation. When asserted, the core writes the data from the page-write buffer to the flash memory address specified by address[n:0]. During a page-write operation, use shift_bytes to shift in the data bytes before asserting page_write.
address[ADDR_WIDTH-1:0]	Input	Flash address to read from, write to, or erase.
datain[7:0]	Input	Parallel 8-bits/1-byte data for page write operations.
rden	Input	Active-high read. While asserted, the core can perform read, fast read, and fast read dual operations.
wren	Input	Active-high write. While asserted, the core can perform write and erase operations. Use with page_write and sector_erase.
shift_bytes	Input	Active-high port that shifts data bytes during a write operation. Use with page_write during page write operations. While shift_bytes is asserted, the core store data on datain[7:0] on the rising edge of clk_in. Shift the required bytes into the flash device until the core finishes storing the data internally.
busy	Output	Indicates that the core is performing a valid operation. Goes high when the core is executing a valid operation; goes low when the operation completes.
dataout[7:0]	Output	Contains the data byte read from flash memory during read operations. This port holds the value of the last data byte read until new a read operation occurs.
data_valid	Output	Indicates that dataout[7:0] contains a valid data byte read from flash memory.
quad_enable	Input	Enables quad mode for the flash. Enabling quad mode disables the WPN and HOLD ports of the flash.
quad_fast_read	Input	Active-high port that executes a fast read quad operation.
quad_page_write	Input	Active-high port that executes a page write quad operation. When asserted, the core writes the data from the page write buffer to the flash memory address specified by address[23:0]. During a page write operation, use shift_bytes to shift in the data bytes before asserting quad_page_write.

Port	Direction	Description
read_deviceID	Input	Active-high port that releases the device from power-down state or read Device ID.
block_erase_64k	Input	Active-high port that executes a block erase operation. When asserted, the core performs a block erase operation on the flash address specified by address[*n*:0].
fourbyte_addr_enable	Input	Enables 4-byte addressing mode.

Table 5: SPI Ports

Port	Direction	Description
miso	Input	Serial data. Bit 0 in fast read dual and quad mode operations.
miso_1	Input	Serial data. Bit 1 in fast read dual and quad mode operations.
miso_2	Input	Serial data. Bit 2 in fast read quad mode operations.
miso_3	Input	Serial data. Bit 3 in fast read quad mode operations.
sclk	Output	SPI clock.
nss	Output	SPI active low chip select.
mosi	Output	Serial data out. Bit 0 for page write, quad page write and erase operations.
mosi_1	Output	Serial data out. Bit 1 quad mode operations.
mosi_2	Output	Serial data out. Bit 2 quad mode operations.
mosi_3	Output	Serial data out. Bit 3 quad mode operations.
mosi_oe	Output	Serial data out enable.

ASMI SPI Flash Controller Core Operations

You must wait for the busy signal to go low before triggering any operation.

> ℹ️ **Note:** The following waveforms are based on operations using default parameter settings.

Page Write Operations

During a page write operation, assert the wren and shift_bytes signals to trigger the core to store the data byte on mosi. The busy signal remains asserted until the busy status bit in the flash device is 0.

Figure 2: Page Write Waveform
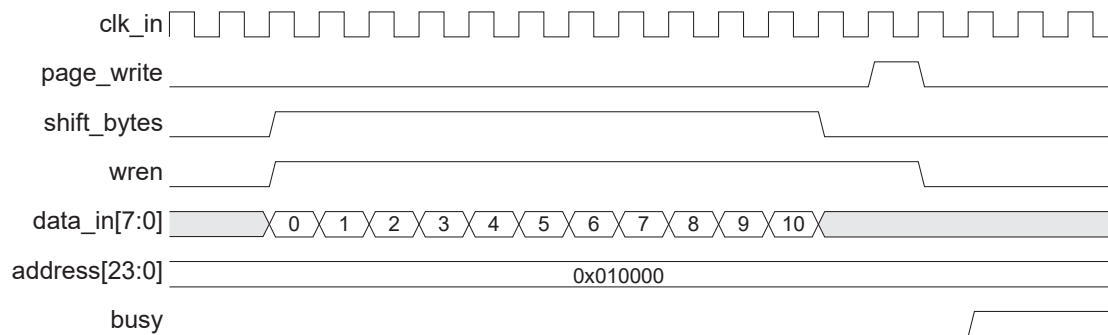
Page write quad operation is similar to page write, except you use the quad_page_write signal to indicate the quad page write operation. For page write quad, assert the quad enable for one clock cycle. Set the page write quad signals after the busy signal is deasserted as shown in the following waveform.
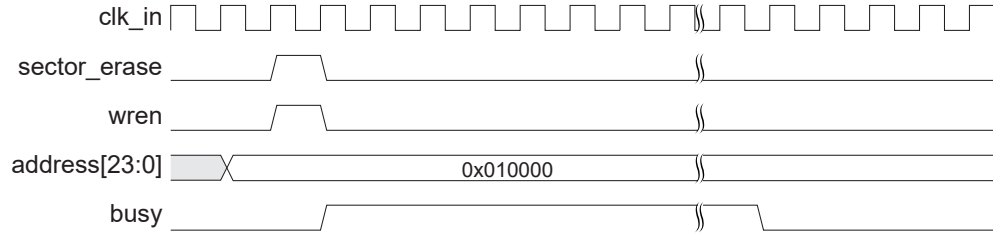
Figure 3: Page Write Quad Waveform
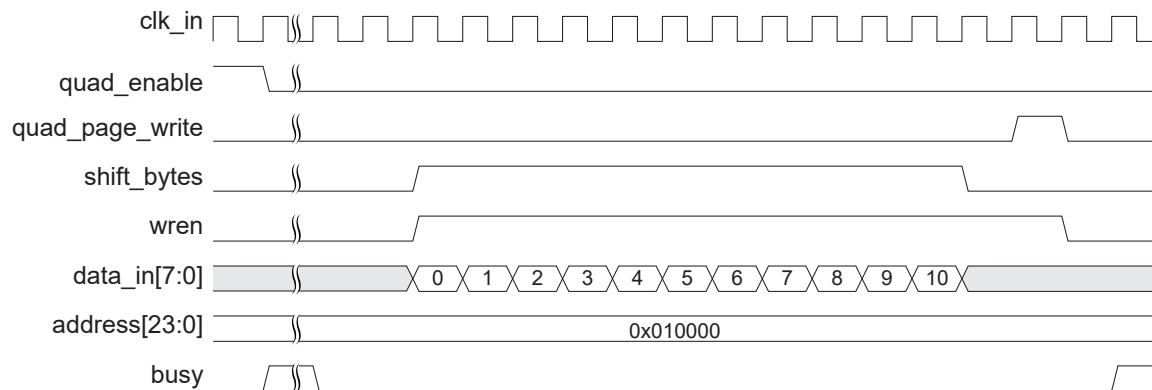
Sector Erase Operation

To initiate a sector erase operation, assert the `wren` and `sector erase` signals. The core asserts the `busy` signal when erasing the sector and de-asserts it when the flash device's `busy` status bit is `0`.

Figure 4: Sector Erase Waveform

Fast Read Operations

> **Note:** All of the following read operation waveforms uses 200 clock cycles for `rden`.

In a fast read operation, use the `fast_read` signal to indicate a fast read operation from the flash device, and assert `rden` to enable the read operation. The read address must appear on `address[23:0]` before `fast_read` goes high. Your RTL should monitor the `data_valid` signal and sample `miso` when `data_valid` is 1.

Figure 5: Fast Read Waveform
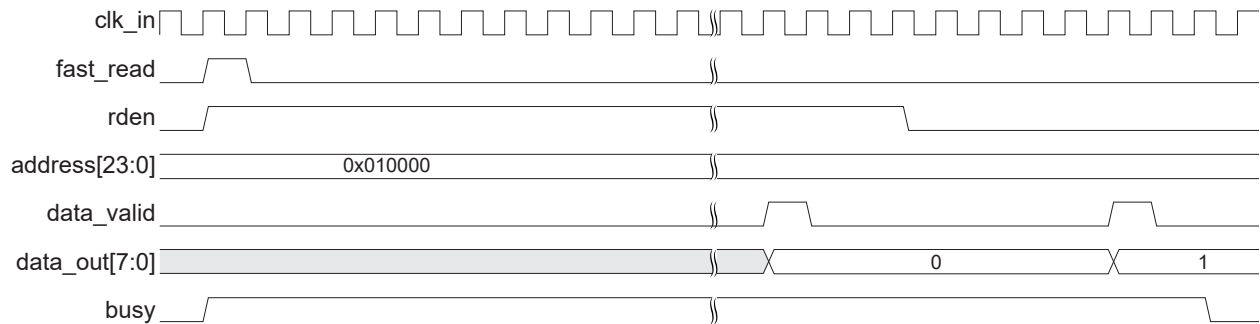
Fast read dual and quad operations are similar to fast read, except you use the `fast_read_dual` or `quad_fast_read` signal to indicate the fast read dual or fast read quad operations respectively.
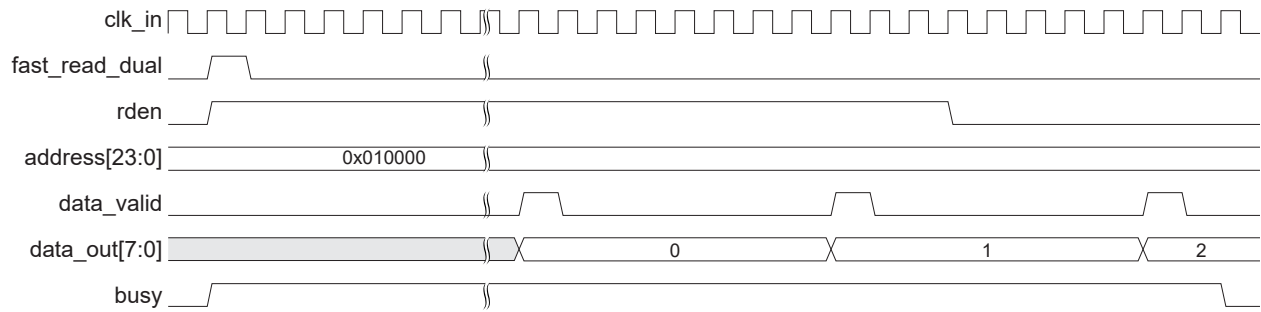
Figure 6: Fast Read Dual Waveform

For fast read quad, assert the `quad enable` for one clock cycle. Set the fast read quad signals after the `busy` signal is deasserted as shown in the following waveform.

Figure 7: Fast Read Quad Waveform
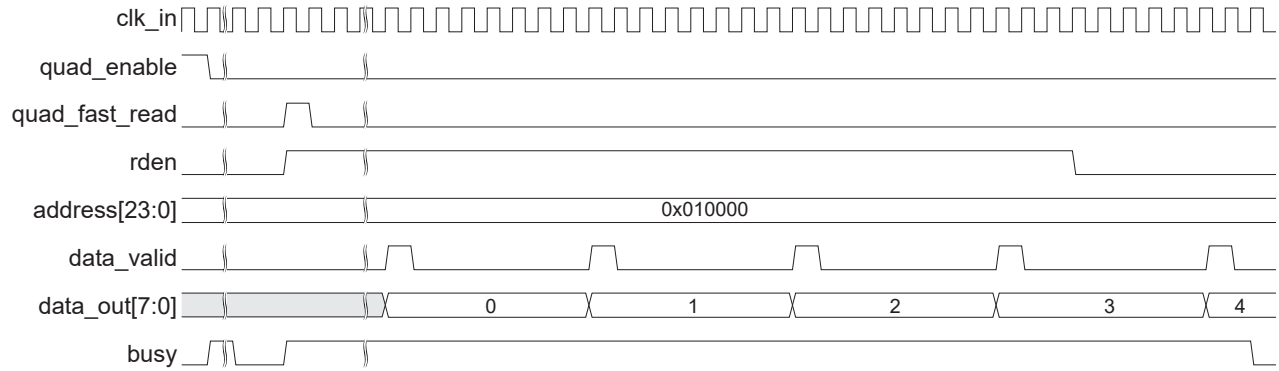
Supported Flash Operation Codes

> **Note:** Refer to your flash device data sheet for more information.

Table 6: Supported Operation Code

Operation	Operation Code
Reset enable	8'h66
Reset device	8'h99
Read device unique ID	8'h4B
Release from power-down or read Device ID	8'hAB
Write	8'h02
Write status register 1	8'h01
Write status register 2	8'h31
Write status register 3	8'h11
Read status register 1	8'h05
Read status register 2	8'h35
Read status register 3	8'h15
Read device identification	8'h9F
Read manufacturing identification	8'h90
Write enable	8'h06
Write enable for volatile status register	8'h50
Write disable	8'h04
Read	8'h03
Fast read	8'h0B
Fast read dual	8'h3B
Fast read quad	8'h6B
Chip erase type 0	8'hC7
Die erase type 1	8'h60
4 KB subsector erase	8'h20
32 KB subsector erase	8'h52
64 KB subsector sector erase	8'hD8
Quad write	8'h32
Write with 4-byte address	8'h12
Read with 4-byte address	8'h13
Fast read with 4-byte address	8'h0C
Fast read dual with 4-byte address	8'h3C
4 KB subsector erase with 4-byte address	8'h21
64 KB subsector sector erase with 4-byte address	8'hDC

Interface Settings

The following table lists the interface connections between the flash device and the ASMI SPI flash controller core. You need to set the GPIO block mode in the Interface Designer as shown in the following tables.

Table 7: Single SPI Mode

Flash Device Data Pin	ASMI SPI Flash Controller SPI Pin	GPIO Block Mode
D0	mosi	output
D1	miso	input

Table 8: Dual SPI Mode

Flash Device Data Pin	ASMI SPI Flash Controller SPI Pin	GPIO Block Mode
D0	mosi	inout
	miso_1	
D1	miso	input

Table 9: Quad SPI Mode

Flash Device Data Pin	ASMI SPI Flash Controller SPI Pin	GPIO Block Mode
D0	mosi	inout
	miso_1	
D1	mosi_1	inout
	miso	
D2	mosi_2	inout
	miso_2	
D3	mosi_3	inout
	miso_3	

IP Manager

The Efinity® IP Manager is an interactive wizard that helps you customize and generate Efinix® IP cores. The IP Manager performs validation checks on the parameters you set to ensure that your selections are valid. When you generate the IP core, you can optionally generate an example design targeting an Efinix development board and/or a testbench. This wizard is helpful in situations in which you use several IP cores, multiple instances of an IP core with different parameters, or the same IP core for different projects.

> ⓘ **Note:** Not all Efinix IP cores include an example design or a testbench.

Generating the ASMI SPI flash controller Core with the IP Manager

The following steps explain how to customize an IP core with the IP Configuration wizard.

1. Open the IP Catalog.
2. Choose **Memory Controllers** > **ASMI SPI flash controller** core and click **Next**. The **IP Configuration** wizard opens.
3. Enter the module name in the **Module Name** box.

> ⓘ **Note:** You cannot generate the core without a module name.

4. Customize the IP core using the options shown in the wizard. For detailed information on the options, refer to the *Customizing the ASMI SPI flash controller* section.
5. (Optional) In the **Deliverables** tab, specify whether to generate an IP core example design targeting an Efinix® development board and/or testbench. These options are turned on by default.
6. (Optional) In the **Summary** tab, review your selections.
7. Click **Generate** to generate the IP core and other selected deliverables.
8. In the **Review configuration generation** dialog box, click **Generate**. The Console in the **Summary** tab shows the generation status.

> ⓘ **Note:** You can disable the **Review configuration generation** dialog box by turning off the **Show Confirmation Box** option in the wizard.

9. When generation finishes, the wizard displays the **Generation Success** dialog box. Click **OK** to close the wizard.

The wizard adds the IP to your project and displays it under **IP** in the Project pane.

Generated Files

The IP Manager generates these files and directories:
- **<module name>_define.vh**—Contains the customized parameters.
- **<module name>_tmpl.v**—Verilog HDL instantiation template.
- **<module name>_tmpl.vhd**—VHDL instantiation template.
- **<module name>.v**—IP source code.
- **settings.json**—Configuration file.
- **<kit name>_devkit**—Has generated RTL, example design, and Efinity® project targeting a specific development board.
- **Testbench**—Contains generated RTL and testbench files.

Customizing the ASMI SPI Flash Controller

The core has parameters so you can customize its function. You set the parameters in the General tab of the core's IP Configuration window.

Table 10: ASMI SPI flash controller Core Parameters

Parameter	Options	Description
Address Width	24 or 32	Set address port bit addressing. Default: 24
Clock Divider for SPI Clock	1 – 20	Sets the clock divider for the SPI clock. The effective frequency is the frequency of clk_in divided by (2 * Clock Divider for SPI Clock). Maximum SPI clock frequency supported is 50 MHz. Default: 2

# ASMI SPI Flash Controller Example Design

You can choose to generate the example design when generating the core in the IP Manager Configuration window. Compile the example design project and download the **.hex** or **.bit** file to your board.

⚠ **Important:** Efinix tested the example design generated with the default parameter options only.

The example designs target the Trion® T20 BGA256 Development Board, Trion T120 BGA324 Development Board, Titanium Ti60 F225 Development Board, and Titanium Ti180 M484 Development Board. The design implements a ASMI SPI flash controller in the FPGA, which allows you to use the ASMI SPI flash controller to access the flash memory via sector erase, write, and read commands. The maximum supported frequency is 50 MHz.

Figure 8: ASMI SPI Flash Controller Example Design
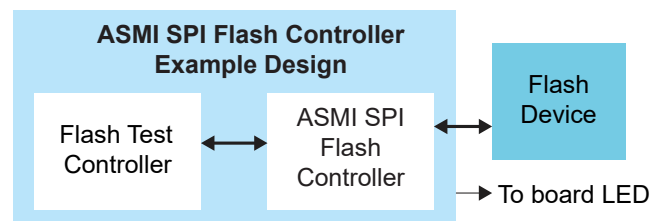
Table 11: Example Design Support

Development Board	Example Design
T20 BGA256	Dual read and write operations using 3-byte addressing mode.
Ti60 F225	
T120 BGA324	Dual read and write, quad read and write operations using 3-byte addressing mode.
Ti180 M484	Dual read and write operations using 3-byte and 4-byte addressing modes.

Table 12: Titanium Example Design Implementation

FPGA	Logic Elements (Logic, adders, Flipflops, etc.)	Memory Blocks	DSP Blocks	f_{MAX} (MHz)	Efinity® Version[2]
Ti60 F225 C4	2,004/60,800 (2.69%)	1/256 (0.39%)	0	231	2022.2
Ti180 M484 C4	1,884/172,800 (1.09%)	1/1,280 (0.08%)	0	224	2022.2

Table 13: Trion® Example Design Implementation

FPGA	LUTs	Registers	Memory Blocks	Multipliers	f_{MAX} (MHz)	Efinity Version[2]
T20 BGA256 C4	1,576	466	2	0	127	2022.2
T120 BGA324 C4	1,629	617	2	0	119	2022.2

T20 BGA256 and Ti60 F225 Development Board Example Design

After the FPGA powers up, the ASMI SPI flash controller performs this sequence:

1. Soft reset.
2. Set address to 0x350000.
3. Sector erase at address 0x350000.
4. Write 8-bit data (0x5A) to address 0x350000.
5. Fast read from address 0x350000.
6. Set address to 0x000004.
7. Sector erase at address 0x000004.
8. Write 8-bit data (0x33) to address 0x000004.
9. Fast read dual from address 0x000004
10. Set address to 0x050000.
11. Release the device from power-down state.
12. Set address to 0x0000ff.
13. Block erase at address 0x0000ff.
14. Write 8-bit data (0x50) to address 0x0000ff.
15. Fast read dual from address 0x0000ff.
16. Set address to 0x000024.
17. Block erase at address 0x000024.
18. Write 8-bit data (0x23) to address 0x000024.
19. Fast read from address 0x000024.
20. When done, the LEDs display:
    - T20 BGA256 Development Board—LEDs D4 and D3 turned on
    - Ti60 F225 Development Board—LED D16 turned on blue and green, and LED D17 turned on red

Figure 9: Titanium Ti60 F225 Development Board LED Outputs

---

[2] Using Verilog HDL.

Ti180 M484 Development Board Example Design

After the FPGA powers up, the ASMI SPI flash controller performs this sequence:

1. Soft reset.
2. Set address to `0x350000`.
3. Sector erase at address `0x350000` (4-byte addressing).
4. Write 8-bit data (`0x5A`) to address `0x350000` (4-byte addressing).
5. Fast read from address `0x350000` (4-byte addressing).
6. Set address to `0x000004`.
7. Sector erase at address `0x000004` (3-byte addressing).
8. Write 8-bit data (`0x33`) to address `0x000004` (3-byte addressing).
9. Fast read dual from address `0x000004` (3-byte addressing).
10. Set address to `0x050000`.
11. Release the device from power down state.
12. Set address to `0x0000ff`.
13. Block erase at address `0x0000ff` (4-byte addressing).
14. Write 8-bit data (`0x50`) to address `0x0000ff` (4-byte addressing).
15. Fast read dual from address `0x0000ff` (4-byte addressing).
16. Set address to `0x000024`.
17. Block erase at address `0x000024` (3-byte addressing).
18. Write 8-bit data (`0x23`) to address `0x000024` (3-byte addressing).
19. Fast read from address `0x000024` (3-byte addressing).
20. When done, the LEDs D2, D3, and D7 turned on.

T120 BGA324 Development Board Example Design

Set the QUAD_EN parameter to 1 in the **flash_test_ctl.v file** to enable quad operations. After the FPGA powers up, the ASMI SPI flash controller performs this sequence:

1. Soft reset.
2. Set address to 0x350000.
3. Sector erase at address 0x350000.
4. Write 8-bit data (0x5A) to address 0x350000.
5. Fast read from address 0x350000.
6. Set address to 0x000004.
7. Sector erase at address 0x000004.
8. Write 8-bit data (0x33) to address 0x000004.
9. Fast read dual from address 0x000004.
10. Set address to 0x050000.
11. Release the device from power down state.
12. Set address to 0x0000ff.
13. Block erase at address 0x0000ff.
14. Write 8-bit data (0x50) to address 0x0000ff.
15. Fast read dual from address 0x0000ff.
16. Set address to 0x000024.
17. Block erase at address 0x000024.
18. Write 8-bit data (0x23) to address 0x000024.
19. Fast read from address 0x000024.
20. Sector erase.
21. Quad page writes 8-bit data (0XC3) to address 0x000010.
22. Read from address 0x000010.
23. Quad read from address 0x000010.
24. When done, the LEDs D7 and D8 turned on.

ASMI SPI Flash Controller Testbench

You can choose to generate the testbench when generating the core in the IP Manager Configuration window.

ⓘ **Note:** You must include all **.v** files generated in the **/testbench** directory in your simulation.

Efinix provides a simulation script for you to run the testbench quickly using the Modelsim software. To run the Modelsim testbench script, run `vsim -do modelsim.do` in a terminal application. You must have Modelsim installed on your computer to use this script.

The testbench performs the erase, write, and read commands to the flash memory. The testbench uses these files:
* **MEM.TXT**—Model for the memory data in the flash device
* **SREG.TXT**—Model for data for flash device status register
* **W25Q256JV.v**—Flash memory model

ⓘ **Note:** The testbench does not support quad operations.

Before running the simulation, uncomment line 17 in the **dbg_defines.v** file to shorten the simulation run time.

After running the write simulation, the test prints the following message:

```
00110011
```

Revision History

Table 14: Revision History

Date	Version	Description
June 2023	3.9	Added Device Support and release notes sections. (DOC-1234) Updated Address Width parameter.
March 2023	3.8	Added support for 4-byte addressing mode and design example targeting Ti180 M484 Development Board. (DOC-1175)
February 2023	3.7	Added note about the resource and performance values in the resource and utilization table are for guidance only.
December 2022	3.6	Added support for 64k block erase and release from power-down state. Updated design example.
August 2022	3.5	Updated for example design and testbench updates in Efinity v2022.1. (DOC-779)
March 2022	3.4	Corrected the example design page write and quad write data.
January 2022	3.3	Updated resource utilization table. (DOC-700)
December 2021	3.2	Updated ASMI SPI Flash Controller operations waveforms.
October 2021	3.1	Added note to state that the f_{MAX} in Resource Utilization and Performance, and Example Design Implementation tables were based on default parameter settings. Updated design example target board to production Titanium Ti60 F225 Development Board and updated Resource Utilization and Performance, and Example Design Implementation tables. (DOC-553)
June 2021	3.0	Added note that all **.v** generated in testbench folder are required for simulation. Updated resource utilization and performance table. Updated example design output and implementation table. Added support for Titanium FPGAs and example design for Titanium Ti60 F225 Development Board. Added support for quad SPI interface.
December 2020	2.0	Updated core name to ASMI SPI Flash Controller core. Updated user guide for Efinix® IP Manager which includes added IP Manager topics, updated parameters, and user guide structure.
May 2020	1.1	Added version 1.1 of the core. Updated the logic utilization and f_{MAX} in resource utilization and performance table. Removed support for 32-bit addressing. Updated SCLK_DIV parameter description. Updated address port description. Updated example design flow. Updated the logic utilization and f_{MAX} in example design implementation table. Updated passing simulation results LED display to 10101100.

Date	Version	Description
April 2020	1.0	Initial release.