SD Host Controller Core User Guide

UG-CORE-SD-HOST-v1.3
February 2023
www.efinixinc.com

Contents

Introduction

SD host controller is a standard memory controller to access Secure Digital (SD) card which is commonly used in embedded devices and consumer electronics. The SD Host Controller core complies with the SD Host Controller Specification v2.00 and Physical Layer Specification v2.0 with 32 bits system bus and built-in ADMA. The module does not support EMMC and SDIO protocols.

Use the IP Manager to select IP, customize it, and generate files. The SD Host 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

- 1-bit, 4-bit SD modes
- SDR12, SDR25 and SDR50 speed modes
- Supports SD card detection
- ADMA transfer and memory-mapped transfer
- Single block and multi-block transfer
- Cyclic redundancy check (CRC) for command and data
- Programmable data buffer
- AXI4-lite interface for AXI4-lite and AXI4 for memory transfer
- Supports interrupt on command and data completion
- Includes example designs targeting the Trion® T120 BGA576 Development Board

FPGA Support

The SD Host Controller core supports all Titanium and Trion® FPGAs.

Resource Utilization and Performance

Note: The resources and performance values provided are just guidance and change depending on the device resource utilization, design congestion, and user design.

Titanium Resource Utilization and Performance

FPGA	Logic and Adders	Flip-flops	Memory Blocks	DSP48 Blocks	f_{MAX} (MHz)[1]	Efinity® Version[2]
Ti60 F225 C4	2,925	3,486	12	0	298	2021.2

Trion® Resource Utilization and Performance

FPGA	Logic Utilization (LUTs)	Registers	Memory Blocks	Multipliers	f_{MAX} (MHz)[1]	Efinity® Version[2]
T120 BGA576 C4	3,243	3,802	26	0	109	2021.1

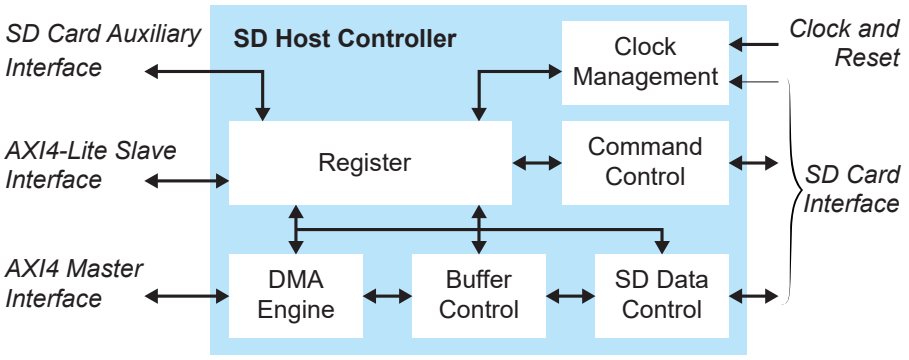
(1) Using default parameter settings.
(2) Using Verilog HDL.

Functional Description

The SD Host Controller core consists of the following modules:

- *Clock Management*—Manages and generates clock source to SD receiver.
- *SD Command Control*—Transits command out from SD host controller and listening response from SD receiver
- *SD Data Control*—Transits data out from SD host controller and receive data coming back from SD receiver
- *Buffer Control*—Manages incoming data traffic and outgoing data traffic.
- *DMA Engine*—Manages ADMA data transfer protocol.
- *Register*—Stores control setting for host controller and status/response from SD receiver.

Figure 1: SD Host Controller PLL Calibration Block Diagram



Ports

Table 1: SD Host Controller Clock and Reset Interface

Name	Direction	Description
sd_clk	Input	SD base clock.
sd_rst	Input	SD base reset.

Table 2: SD Card Interface

Name	Direction	Description
sd_clk_o	Output	SD card clock.
sd_cmd_i	Input	SD card command input.
sd_cmd_o	Output	SD card command output
sd_cmd_oe	Output	SD card command output enable.
sd_dat_i [3:0]	Input	SD card data input.
sd_dat_o [3:0]	Output	SD card data output.
sd_dat_oe	Output	SD card data Output output enable.

Table 3: SD Card Auxiliary Signal Interface

Name	Direction	Description
sd_int	Output	SD interrupt.
sd_wp	Input	SD write protect switch.
sd_cd_n	Input	SD card not present.

Table 4: AXI4-Lite Slave Register Interface

Name	Direction	Description
s_axi_aclk	Input	AXI4-Lite interface clock.
s_axi_awaddr [9:0]	Input	AXI4-Lite write address bus.
s_axi_awvalid	Input	AXI4-Lite write address valid strobe.
s_axi_awready	Output	AXI4-Lite write address ready signal.
s_axi_wdata [31:0]	Input	AXI4-Lite write data.
s_axi_wvalid	Input	AXI4-Lite write data valid strobe.
s_axi_wready	Output	AXI4-Lite write ready signal.
s_axi_bresp [1:0]	Output	AXI4-Lite write response.
s_axi_bvalid	Output	AXI4-Lite write response valid strobe.
s_axi_bready	Input	AXI4-Lite write response ready signal.
s_axi_araddr [9:0]	Input	AXI4-Lite read address bus.
s_axi_arvalid	Input	AXI4-Lite read address valid strobe.
s_axi_arready	Output	AXI4-Lite read address ready signal.
s_axi_rdata [31:0]	Output	AXI4-Lite read data.
s_axi_rresp [1:0]	Output	AXI4-Lite read response.
s_axi_rvalid	Output	AXI4-Lite read data valid strobe.
s_axi_rready	Input	AXI4-Lite read data ready signal.

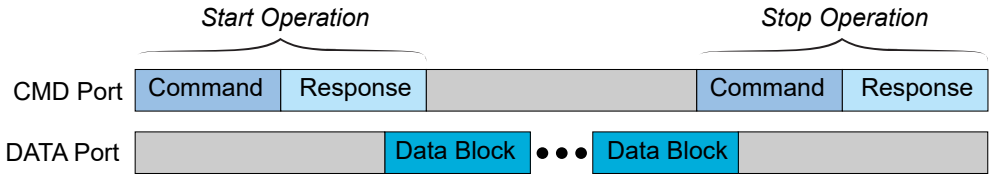
Table 5: AXI4 Master Memory Interface

Name	Direction	Description
m_axi_clk	Input	AXI4 interface clock.
m_axi_awaddr [31:0]	Output	AXI4 write address.
m_axi_awlen [7:0]	Output	AXI4 write burst length.
m_axi_awsize [2:0]	Output	AXI4 write burst size.
m_axi_awburst [1:0]	Output	AXI4 write burst type.
m_axi_awlock [1:0]	Output	AXI4 write lock type.
m_axi_awcache [3:0]	Output	AXI4 write cache type.
m_axi_awprot [2:0]	Output	AXI4 write protection type.
m_axi_awvalid	Output	AXI4 write address valid.
m_axi_awready	Input	AXI4 write address ready.
m_axi_wdata [31:0]	Output	AXI4 write data.
m_axi_wstrb [3:0]	Output	AXI4 write strobes.
m_axi_wlast	Output	AXI4 write last.
m_axi_wvalid	Output	AXI4 write valid.
m_axi_wready	Input	AXI4 write ready.
m_axi_bresp [1:0]	Input	AXI4 write response.
m_axi_bvalid	Input	AXI4 write response valid.
m_axi_bready	Output	AXI4 write response ready.
m_axi_araddr [31:0]	Output	AXI4 read address.
m_axi_arlen [7:0]	Output	AXI4 read burst length.
m_axi_arsize [2:0]	Output	AXI4 read burst size.
m_axi_arburst [1:0]	Output	AXI4 read burst type.
m_axi_arlock [1:0]	Output	AXI4 read lock type.
m_axi_arcache [3:0]	Output	AXI4 read cache type.
m_axi_arprot [2:0]	Output	AXI4 read protection type.
m_axi_arvalid	Output	AXI4 read address valid.
m_axi_arready	Input	AXI4 read address ready.
m_axi_rdata [31:0]	Input	AXI4 read data.
m_axi_rresp [1:0]	Input	AXI4 read response.
m_axi_rlast	Input	AXI4 read last.
m_axi_rvalid	Input	AXI4 read valid.
m_axi_rready	Output	AXI4 read ready.

SD Host Operation

A command is a token that triggers an operation. A response is a token sent from an addressed card by answering the request from the SD host controller. The commands and responses are transferred through the CMD port. When the communication is formed and confirmed at both ends, the data can be transferred from the card to the SD host controller or vice versa via DATA ports. The SD host controller uses command tokens to communicate with the SD card and wait for the response from the SD card. Then the SD host controller transfers or receives streams of data from or to the SD card.

Figure 2: SD Host Controller Operation



Initialization

During power-on, the SD Host Controller core must go through the initialization and identification process before transferring any data. This is mainly due to the SD Host Controller core needing to retrieve the SD card's basic information and ensure the card is ready to be accessed.

Block Read and Multiple Blocks Read

The SD Host Controller core supports single and multiple blocks read. A basic unit of a block has a maximum size of 512 bytes. Send a `CMD17` command to let the SD card know it is a single block read. For multiple blocks read, send a `CMD18` command as a start indicator, followed by receiving consecutive data blocks, and end it by issuing `CMD12` command to indicate a stop transmission.

Block Write and Multiple Blocks Write

The SD Host Controller core supports single and multiple blocks write. A basic unit of a block has a maximum size of 512 bytes. Send a `CMD24` command to let the SD card know it is a single block write. For multiple blocks write, send a `CMD25` command as a start indicator, followed by writing consecutive data blocks, and end it by issuing `CMD12` command to indicate a stop transmission.

ADMA Transfer

The SD Host Controller core supports advanced direct memory access (ADMA) data transfer. This transfer mode is usually executed when you transfer a large amount of data multiple times to or from the SD card.

You need to create a descriptor list and store it in system memory. The maximum transfer for each descriptor is 65536 bytes, including a target address. If you want to transfer more than that, you need to create multiple descriptors. After creating the descriptor list, start the ADMA, and the SD host automatically fetches the descriptor from system memory and starts the data transfer according to the list. The last descriptor must contain a stop bit information to indicate that it is the final data transfer. The ADMA interrupts the CPU once the final descriptor is executed and the transfer is completed.

Register Definition

Table 6: Register Description

Address Offset	Name	R/W
0x000	Version	RO
0x004	Base Register 0	R/W
0x008	Base Status Register 0	RO
0x100	Argument 2[31:0]	R/W
0x104	Block Size Register Block Count Register	R/W
0x108	Argument 1[31:0]	R/W
0x10C	Transfer Mode Register Command Register	R/W
0x110	Command_Response [31:0]	RO
0x114	Command_Response [63:32]	RO
0x118	Command_Response [95:64]	RO
0x11c	Command_Response [127:96]	RO
0x120	Buffer Data Port Register	R/W
0x124	Present State Register	R/W
0x128	Host Control 1 Register Power Control Register Block Gap Control Register Wakeup Control Register	R/W
0x12c	Clock Control Register Timeout Control Register Software Reset Register	R/W
0x130	Normal Interrupt Status Register Error Interrupt Status Register	R/W
0x134	Normal Interrupt Status Enable Register Error Interrupt Status Enable Register	R/W
0x138	Normal Interrupt Status Enable Register Error Interrupt Status Enable Register	R/W
0x158	ADMA System Address Register (Lower Word)	R/W
0x15C	ADMA System Address Register (Upper Word)	R/W

Table 7: Base Register 0 (0x004)

Bits	I/O	Description	R/W
31-17	-	Reserved.	-
16	Input	Clock Enable.	R/W
15-0	Input	Clock divider factor.	R/W

Table 8: Base Status Register 0 (0x008)

Bits	I/O	Description	R/W
31-3	-	Reserved.	-
2	Output	SD Card detection: 0: Detected 1: Not Detected	RO
1	Output	Data line is busy: 0: Not Busy 1: Busy	RO
0	Output	Command line is busy: 0: Not Busy 1: Busy	RO

Table 9: Argument 2 Register (0x100)

Bits	I/O	Description	R/W
31-0	Input	Argument 2. Contains the physical system memory address used for ADMA transfers.	R/W

Table 10: Block Size Register (0x104)

Bits	I/O	Description	R/W
15	-	Reserved.	R/W
14-12	Input	Host ADMA Buffer Boundary. Specifies the size of contiguous buffer in the system memory. The ADMA transfer waits at every boundary specified by these fields and the host controller generates the ADMA Interrupt to request the host driver to update the ADMA System Address register. 000b: 4 KB 001b: 8 KB 010b: 16 KB 011b: 32 KB 100b: 64 KB 101b: 128 KB 110b: 256 KB 111b: 512 KB	R/W
11-0	Input	Transfer Block Size. Specifies the block size of data transfers for CMD17, CMD18, CMD24, CMD25, and CMD53. Values ranging from 1 up to the maximum buffer size can be set. For memory, set to 512 bytes. 0000h: No data transfer 0001h: 1 byte 0002h: 2 bytes ... 0200h: 512 bytes ... 0800h: 2048 bytes	R/W

Table 11: Block Count Register (0x104)

Bits	I/O	Description	R/W
31-16	Input	Blocks Count for Current Transfer. Enabled when Block Count Enable in the Transfer Mode register is set to 1 and is valid only for multiple block transfers. The host driver sets this register to a value between 1 and the maximum block count. 0000h: Stop Count 0001h: 1 block 0002h: 2 blocks ... FFFFh: 65535 blocks	R/W

Table 12: Argument 1 Register (0x108)

Bits	I/O	Description	R/W
31-0	Input	Command Argument 1. The SD command argument is specified as bit 39-8 of Command-Format in the Physical Layer Specification.	R/W

Table 13: Transfer Mode Register (0x10C)

Bits	I/O	Description	R/W
16-5	Input	Reserved	R/W
5	Input	Multi/Single Block Select. Set when issuing multiple-block transfer commands using DATA line. 0: Single Block 1: Multiple Block	R/W
4	Input	Data Transfer Direction Select. Defines the direction of DAT line data transfers. 0: Write (Host to Card) 1: Read (Card to Host)	R/W
3-2	Input	Auto CMD Enable. Sets the auto command functions. 00b: Auto Command Disabled 01b: Auto CMD12 Enable 10b: Auto CMD23 Enable 11b: Reserved	R/W
1	Input	Block Count Enable Enables the Block Count register, which is only relevant for multiple block transfers. If ADMA data transfer is more than 65535 blocks, this bit shall be set to 0. In this case, data transfer length is designated by the descriptor table. 0: Disable 1: Enable	R/W
0	Input	DMA Enable. Enables the DMA functionality. 0: No data transfer or Non DMA data transfer 1: DMA Data transfer	

Table 14: Command Register (0x10C)

Bits	I/O	Description	R/W
31-30	-	Reserved.	-
29-24	Input	Command Index. Set to the command number (CMD0-63, ACMD0-63).	R/W
23-22	Input	Command Type. There are three types of special commands: Suspend, Resume and Abort. These bits shall be set to 00b for all other commands. 00b: Other commands 01b: CMD52 for writing "Bus Suspend" in CCCR 10b: CMD52 for writing "Function Select" in CCCR 11b: CMD12, CMD52 for writing "I/O Abort" in CCCR	R/W
21	Input	Data Present Select. Set to 1 to indicate that data is present.	R/W
20	Input	Command Index Check Enable. Set to 1 for the host controller to check the index field in the response to see if it has the same value as the command index.	R/W
19	Input	Command CRC Check Enable. Set to 1 for the host controller to check the CRC field in the response. If an error is detected, it is reported as a Command CRC Error.	R/W
18	-	Reserved.	-
17-16	Input	Response Type Select. 00b: No Response 01b: Response Length 136 10b: Response Length 48 11b: Response Length 48 check Busy after response	R/W

Table 15: Command Response Register (0x110 – 0x11C)

Offset	I/O	Description	R/W
0x110	Output	Command Response 0 – 31.	RO
0x114	Output	Command Response 63 – 32.	RO
0x118	Output	Command Response 95 – 64.	RO
0x11C	Output	Command Response 127 – 96.	RO

Table 16: Buffer Data Port Register (0x120)

Bits	I/O	Description	R/W
31-0	Input	32-bit data port register to access internal buffer.	R/W

Table 17: Present State Register (0x124)

Bits	I/O	Description	R/W
15-12	-	Reserved.	-
11	Output	Buffer Read Enable. Used for non-DMA read transfers. 0: Read Disable 1: Read Enable	RO
10	Output	Buffer Write Enable. Used for non-DMA write transfers. 0: Write Disable 1: Write Enable	RO
9	Output	Read Transfer Active. Indicates completion of a read transfer. 0: No valid data 1: Transferring data	RO
8	Output	Write Transfer Active. Indicates a write transfer is active. 0: No valid data 1: Transferring data	RO
7-4	-	Reserved.	-
3	Input	Re-Tuning Request. Not supported.	R/W
2	Output	DAT Line Active. Indicates whether one of the DAT line on SD Bus is in use. 0: DAT Line Inactive 1: DAT Line Active	RO
1	Output	Command Inhibit (DAT). Indicates if either the DAT Line Active or the Read Transfer Active is set to 1. 0: Can issue command which uses the DAT line 1: Cannot issue command which uses the DAT line	RO
0	Output	Command Inhibit (CMD). Indicates that the CMD line is not in use and the host controller can issue a SD Command using the CMD line. 0: Can issue command using only CMD line 1: Cannot issue command	RO

Table 18: Host Control 1 Register (0x128)

Bits	I/O	Description	R/W
7-2	-	Reserved.	-
1	Input	Data Transfer Width. Selects the data width of the host controller.	R/W
0	Input	LED Control. Used to caution the user not to remove the card while the SD card is being accessed. 0: LED off 1: LED on	R/W

Table 19: Power Control Register (0x128)

Bits	I/O	Description	R/W
15-8	-	Reserved.	-

Table 20: Block Gap Control Register (0x128)

Bits	I/O	Description	R/W
23-20	-	Reserved.	-
19	Input	Interrupt At Block Gap. Enables interrupt detection at the block gap for a multiple block transfer. 0: Disable 1: Enable	R/W
18	Input	Read Wait Control. If the card supports read wait, set this bit to enable use of the read wait protocol to stop read data using the DAT[2] line. 0: Enable Read Wait Control 1: Disable Read Wait Control	R/W
17	Input	Continue Request. Restart a transaction, which was stopped by the Stop At Block Gap Request. 0: Not affect 1: Restart	R/W
16	Input	Stop At Block Gap Request. Stop executing read and write transaction at the next block gap for non-DMA, SDMA and ADMA transfers 0: Transfer 1: Stop	R/W

Table 21: Wake up Control Register (0x128)

Bits	I/O	Description	R/W
7-0	-	Reserved.	-

Table 22: Clock Control, Timeout and Software Register (0x12C)

Bits	I/O	Description	R/W
31-0	-	Reserved.	-

Table 23: Normal Interrupt Status Register (0x130)

Bits	I/O	Description	R/W
15-9	-	Reserved.	-
8	Output	Card Interrupt. 0: No interrupt 1: Interrupt Detected	RO
7	Output	Card Removal. 0: No interrupt 1: Interrupt Detected	RO
6	Output	Card Insertion. 0: No interrupt 1: Interrupt Detected	RO
5	Output	Buffer Read Ready. 0: No interrupt 1: Interrupt Detected	RO
4	Output	Buffer Write Ready. 0: No interrupt 1: Interrupt Detected	RO
3	-	Reserved	-
2	Output	Block Gap Event. 0: No interrupt 1: Interrupt Detected	RO
1	Output	Transfer Complete. 0: No interrupt 1: Interrupt Detected	RO
0	Output	Command Complete. 0: No interrupt 1: Interrupt Detected	RO

Table 24: Error Interrupt Status Register (0x130)

Bits	I/O	Description	R/W
31-22	-	Reserved.	-
21	Output	Data CRC Error. 0: No interrupt 1: Interrupt Detected	RO
20	-	Reserved.	-
19	Output	Command Index Error. 0: No interrupt 1: Interrupt Detected	RO
18	Output	Command End Bit Error. 0: No interrupt 1: Interrupt Detected	RO
17	Output	Command CRC Error. 0: No interrupt 1: Interrupt Detected	RO
16	Output	Command Timeout Error. 0: No interrupt 1: Interrupt Detected	RO

Table 25: Normal Interrupt Status Enable Register (0x134)

Bits	I/O	Description	R/W
15-9	-	Reserved.	-
8	Input	Card Interrupt. 0: No interrupt 1: Interrupt Detected	R/W
7	Input	Card Removal. 0: No interrupt 1: Interrupt Detected	R/W
6	Input	Card Insertion. 0: No interrupt 1: Interrupt Detected	R/W
5	Input	Buffer Read Ready. 0: No interrupt 1: Interrupt Detected	R/W
4	Input	Buffer Write Ready. 0: No interrupt 1: Interrupt Detected	R/W
3	-	Reserved	-
2	Input	Block Gap Event. 0: No interrupt 1: Interrupt Detected	R/W
1	Input	Transfer Complete. 0: No interrupt 1: Interrupt Detected	R/W
0	Input	Command Complete. 0: No interrupt 1: Interrupt Detected	R/W

Table 26: Error Interrupt Status Enable Register (0x134)

Bits	I/O	Description	R/W
31-22	-	Reserved.	-
21	Input	Data CRC Error. 0: No interrupt 1: Enable interrupt	R/W
20	-	Reserved.	-
19	Input	Command Index Error. 0: No interrupt 1: Enable interrupt	R/W
18	Input	Command End Bit Error. 0: No interrupt 1: Enable interrupt	R/W
17	Input	Command CRC Error. 0: No interrupt 1: Enable interrupt	R/W
16	Input	Command Timeout Error. 0: No interrupt 1: Enable interrupt	R/W

Table 27: Normal Interrupt Signal Enable Register (0x138)

Bits	I/O	Description	R/W
15-9	-	Reserved.	-
8	Input	Card Interrupt. 0: Masked 1: Enabled	R/W
7	Input	Card Removal. 0: Masked 1: Enabled	R/W
6	Input	Card Insertion. 0: Masked 1: Enabled	R/W
5	Input	Buffer Read Ready. 0: Masked 1: Enabled	R/W
4	Input	Buffer Write Ready. 0: Masked 1: Enabled	R/W
3	-	Reserved.	-
2	Input	Block Gap Event. 0: Masked 1: Enabled	R/W
1	Input	Transfer Complete. 0: Masked 1: Enabled	R/W
0	Input	Command Complete. 0: Masked 1: Enabled	R/W

Table 28: Error Interrupt Signal Enable Register (0x138)

Bits	I/O	Description	R/W
31-22	-	Reserved.	-
21	Input	Data CRC Error. 0: Masked 1: Enabled	R/W
20	-	Reserved.	-
19	Input	Command Index Error. 0: Masked 1: Enabled	R/W
18	Input	Command End Bit Error. 0: Masked 1: Enabled	R/W
17	Input	Command CRC Error. 0: Masked 1: Enabled	R/W
16	Input	Command Timeout Error. 0: Masked 1: Enabled	R/W

Table 29: ADMA System Address Register (0x158)

Bits	I/O	Description	R/W
31-0	-	ADMA System Address (Lower Word). Holds byte address of executing command of the descriptor table.	R/W

Table 30: ADMA System Address Register (0x15C)

Bits	I/O	Description	R/W
31-0	-	ADMA System Address (Upper Word). Holds byte address of executing command of the descriptor table.	R/W

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 a 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 an IP 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 IP core's user guide or on-line help.
5. (Optional) In the **Deliverables** tab, specify whether to generate an IP core example design targeting an Efinix® development board and/or testbench. For SoCs, you can also optionally generate embedded software example code. 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.

Note: Refer to the IP Manager chapter of the Efinity® Software User Guide for more information about the Efinity® IP Manager.

Customizing the SD Host 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 31: SD Host Controller Core Parameters (Memory Tab)

Parameter	Options	Description
Data Buffer Depth	512, 1024, 1024, 2048, 4096	FIFO depth for both SD data transmitter and receiver. Default: 512

SD Host Controller Example Design

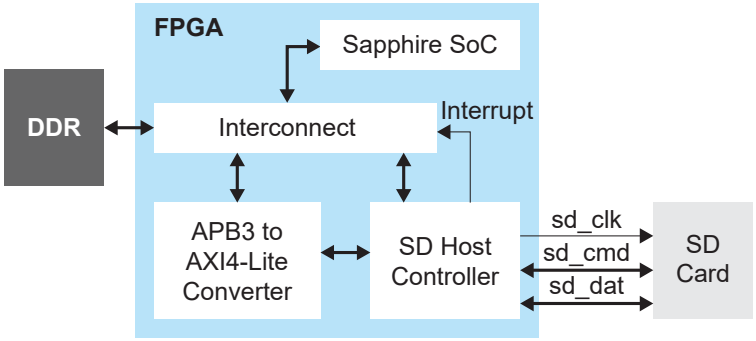
You can choose to generate the example design when generating the core in the IP Manager Configuration window. Compile the example design project and download the **.hex** or **.bit** file to your board.

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

The example designs target the Trion® T120 BGA576 Development Board. The example design consists of a RISC-V SoC and the SD Host Controller core. The design requires an additional SD daughter card attached to the Trion® T120 BGA576 Development Board.

Note: Efinix uses PmodSD from Digilent as the SD daughter card for the example design.

Figure 3: Standalone Example Design Block Diagram



Run the example code, **sdhc.c** in the **embedded_sw** folder to check the multi block read and write operation is working as expected. The UART terminal prints the following message if test is successful:

```
--- EFX-SD Card Demo ---

Initialize...Done

**************START SPEED TEST*******************
**SD CLOCK SPEED = 50000
**CARD SPEED = 25000 kHz
**CARD SIZE = 7580 Mbyte Total BLOCK = 15523840
**SD BUS WIDTH = 4
**BLOCK SIZE = 512 BUFFER OF BLOCK = 64
**TEST SIZE = 32 kbyte
*********************************************

!!!!Warning it will crash the SD card data!!!!

      ###Push Any Key to Continue###
Tested Block 0/15523840          Write s=629 KByte/s   Read s=514 KByte/s
Tested Block 1024/15523840    Write s=629 KByte/s   Read s=514 KByte/s
```

Table 32: Example Design Implementation

FPGA	LUTs	Registers	Memory Blocks	Multipliers	f_{MAX} (MHz)[3]		Efinity® Version[4]
					System Clock	SD Host Clock	
T120 BGA576 C4	8,889	9,407	89	4	62	109	2021.1

SD Host 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 in your computer to use this script.

The example design includes a simulation testbench, **tb_top.v**, which simulates the example design. To run simulation testbench, run `./run.bat` for Windows or `./run.sh` for Linux.

[3] Using default parameter settings.
[4] Using Verilog HDL.

Revision History

Table 33: Revision History

Date	Version	Description
February 2023	1.3	Added note about the resource and performance values in the resource and utilization table are for guidance only.
January 2022	1.2	Updated resource utilization table. (DOC-700)
October 2021	1.1	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	1.0	Initial release.