SD Flash Controller

January 2010 **Reference Design RD1048**

Introduction

The Secure Digital (SD) Card is a Flash-based memory card that is widely used in today's consumer electronic devices. Its high-capacity, compact package and security make it a suitable candidate for both video and audio applications in electronic products. Some common consumer products that make use of SD cards include digital cameras, personal computers, printers, mobile phones and car navigation systems. The SD technology is considered a de-facto standard in the industry for data storage. There is a high demand for SD cards in both portable and desktop consumer products.

The SD card supports two communication transfer protocols, SD Bus mode and SPI Bus mode. The SD Flash Controller Reference Design supports SPI mode. The SPI Communication Protocol complies with SD Physical Layer Simplified Specification 2.0.

This document discusses the following topics.

- Operation command token, response command token and data token
- SD card initialization
- Read and write of a single block on a SD card
- State machine and functional description of design modules

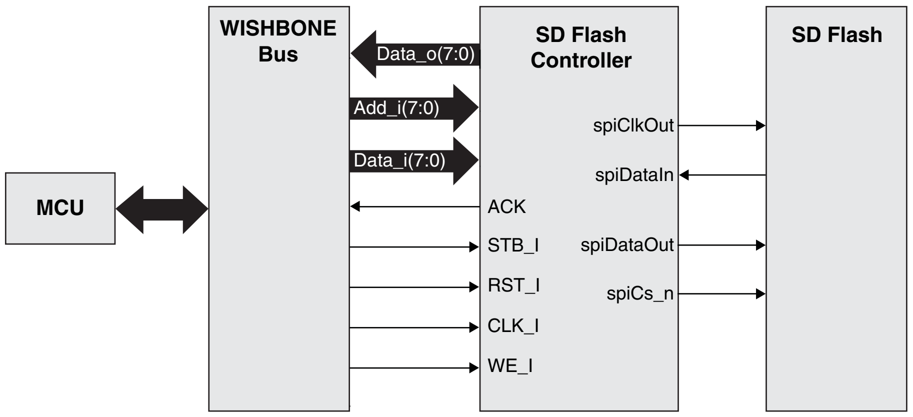
The interface of the SD Flash Controller is based on the WISHBONE bus standard. The WISHBONE interconnection makes system-on-chip (SoC) and design reuse easy by creating a standard data exchange protocol. The SD Flash Controller with WISHBONE bus interface allows the designer to easily connect SoC systems to a standard SD card. It includes reset, initialization, reading of a single block and writing of a single block.

Theory of Operation

Overview

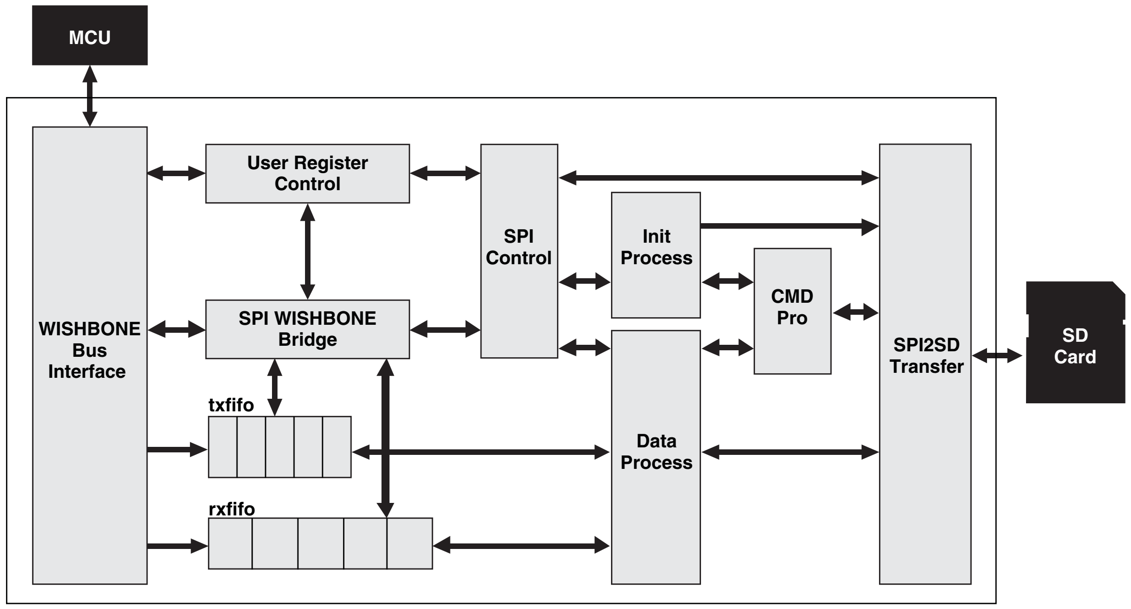
The SD Flash Controller is used to set up communication between a microprocessor and a SD card. It supports the translation protocol from a serial peripheral interface (SPI) bus to the SD stack. Figure 1 shows the signal interface of the system.

Figure 1. System Signal Interface



The block diagram of the SD card controller and signal flow are shown in Figure 2.

Figure 2. SD Card Controller Block Diagram and Signal Flow



System Features

- Full SD memory card support, including card reset, initialization, direct access, single block read and single block write
- Basic SPI mode bus access
- 512-byte receive and transmit FIFO
- 8-bit slave WISHBONE interface
- Data access up to 24 Mbps
- Separate clocks for the WISHBONE interface and SPI core logic
- SPI clock frequency configurable via bus interface
- Data transfer at speeds close to SD card maximum rate
- Force transmit FIFO empty and receive FIFO empty
- Turn off transmit FIFO writing clock and receive FIFO reading clock
- Initialization error indication, writing data error indication, and reading data error indication

Signal Descriptions

A summary of the top-level signals is shown in Table 1.

Table 1. Top-level Signal Descriptions

Interface	Signal	Type	Description
Microprocessor Interface	spiSysClk	Input	Main process clock, 50MHz
	clk_i	Input	WISHBONE clock input, asynchronous with spiSysClk_25MHz
	rst_i	Input	WISHBONE reset. Synchronous to clk_i, hardware reset signal.
	address_i[7:0]	Input	WISHBONE address input signal
	data_i[7:0]	Input	WISHBONE data input signal
	data_o[7:0]	Output	WISHBONE data output signal
	writeEn	Input	WISHBONE write enable signal
	strobe_i	Input	WISHBONE strobe input signal
	ack_o	Output	WISHBONE acknowledge signal
SD Card Interface	spiClkOut	Output	SPI clock signal, clock speed configurable
	spiDataIn	Input	SPI serial data from slave
	spiDataOut	Output	SPI serial data to slave
	spiCs_n	Output	SPI device chip select

Operation

The SD Flash Controller includes card-specific functions composed of reset, initialization, reading single block and writing single block, and implements the standard interface to card stack as the bus master for the SD system. The SD Flash Controller is a slave to the Microprocessor Unit (MCU) and consists of command and control registers, transmits FIFOs and receives FIFOs. The host has access to these registers and FIFOs and generates commands, interprets responses, and controls subsequent actions. The SPI bus connects the card to the controller. The host can turn spiClkOut on and off. The card stack and the controller communicate serially through the data lines and implement a message-based protocol. The messages consist of the following tokens:

- **Command Token** – A command token is 6 bytes long. The command set includes card initialization, card register read and write, data transfers and data erasure. The SD Flash Controller sends command token serially on the spiDataIn signal. The format of a command token is shown in Table 2.

Table 2. Command Token Format

Bit Position	47	46	[45:40]	[39:8]	[7:1]	0
Width (bits)	1	1	6	32	7	1
Value	0	1	x	x	x	1
Description	Start bit	Transmission bit	Command index	Argument	crc7	End bit

- **Response Token** – A response token is an answer to a command token. Each command has either a specific response type or no response type. The format of a response token varies according to the expected response and the card's mode. The response token formats are detailed in the SD Physical Layer Specification. The response token is transferred on the spiDataOut wire.
- **Data/Data Token** – The data token is transferred serially between the host and the card in 8-bit blocks at rates up to 24 Mbps for SPI mode data transfers. In this reference design, each data block is 512 bytes which equals one sector of the SD card. Before the valid data is transmitted, the host must transmit the data token to the SD card. The format of the data token depends on the card's mode. Table 3 shows the data token format for both SD mode and SPI mode. The data token is transferred on the spiDataIn wire.

Table 3. Data Token Format

Stream Data	1	x	No CRC	1
Block Data	0	x	x	1
Description	Start bit	Data	CRC7	End bit

SPI Mode Description

SPI mode is an optional communications protocol of the SD card. SPI mode is selected on power-up when the first reset command (CMD0) is sent and the chip is selected. The communication mode cannot be changed unless the SD card goes through power cycling.

In SPI mode, all command, data, and response tokens are 8 bits long and are transmitted immediately following the assertion of the respective chip select. The SD samples input data on the positive edge of spiClkOut and output data on the negative edge of the spiClkOut. The command token is protected with a 7-bit CRC. The SD card sends a response to the controller after a command token. The response token has four formats, including CRC results. The length of the response tokens is one, two, or five bytes. In this reference design, the results of CRC calculation are appended in the command and data.

Read and write data transfers are protected with the 16-bit CRC. In write data transfers, after the data and 16-bit CRC have been transmitted, the card sends a 5-bit status token. The CRC status token indicates if the data transmission was erroneous. After the CRC status token, the card can indicate that it is busy programming data by pulling the data line low.

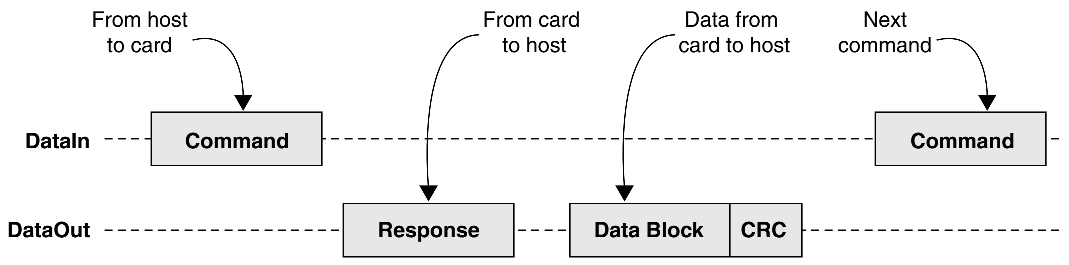
SPI Bus Description

The SD card communication channel is serial in nature and is initiated by a start bit and terminated by a stop bit. The SPI communication format is byte oriented. Every command or data block is built of 8-bit bytes and is byte-aligned to the chip select (CS) signal. All communication between host and cards is controlled by the host. The host starts every bus transaction by asserting the CS signal low.

SPI Read Data Operation

SPI mode supports single block read and multi-block read. The host can send CMD17 or CMD18 to read SD card data. When a reading operation is received by the SD card, the card generates a response token in response to the reading operation. Figure 3 shows the SPI mode read timing diagram.

Figure 3. Single Block Read Operation

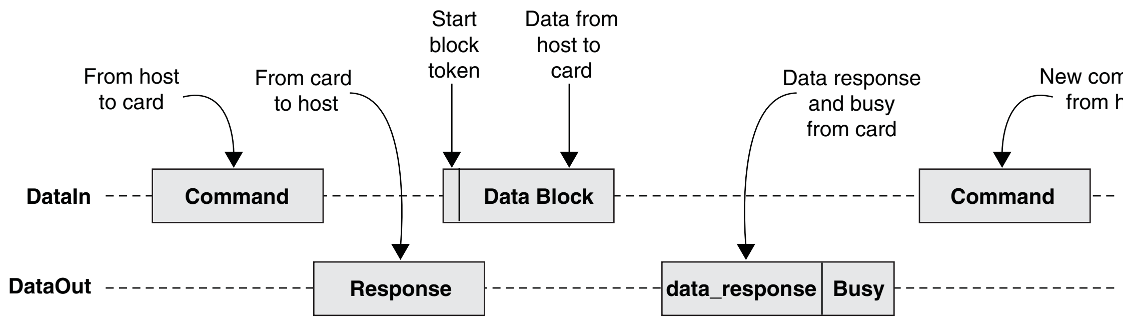


The data block is protected by CRC16 which is complied with the CCITT ($X^{16}+X^{12}+X^5+1$) standard. The address of the read operation is the parameter of CMD17, which must include a legal physical sector address. If the operation generates an error, the SD card generates an error response to the controller. The controller judges the response indication to determine what to do in the next step. When the time of the operation is longer than the time specified in the SD specification, the SD card generates a time out feedback signal as the response signal. In this reference design, the command is sent as a constant parameter to the SD card.

SPI Write Data Operation

SPI mode supports single block write operations and multi-block write operations. The host sends CMD24 or CMD25 to write to the SD card. When a writing operation is requested by the host, the SD card generates a response token to the writing operation which indicates the state of the SD card. Figure 4 shows SPI mode write timing diagrams.

Figure 4. Single Block Writing Operation



Each data block has a data token as the indication to the SD card. When a block is received, the SD card generates a response. If the data block is correctly received and the CRC results are right, the transmitted data is programmed into the sector by the SD card. In return, the SD card generates a busy signal as a response. The address parameter is part of the write command.

Reset Operation

The SD Flash controller can only be reset by a hard or soft reset from the microprocessor. The hard reset is from the hardware reset. All registers and FIFO controls are set to their default values after any reset. The user can send the CMD0 to reset the SD card. The CMD0 command can break any operation even if the SD card is busy.

Initialization Operation

After reset, the SD card must be initialized by sending more than 80 clocks to it on the SpiClkOut signal. To initialize the SD card, set the DAT [INIT] bit to a 1. This sends 80 clocks before the current command in the CMD register. This function is useful for acquiring new cards that have been inserted on the bus. Chip selects are not asserted during the initialization sequence while in SPI mode.

After the 80-clock initialization sequence, the software must continuously send CMD1 by loading the appropriate command index into the SD_CMD register until the card indicates that the power-up sequence is complete. The MCU can then assign an address to the card or put it into SPI mode.

Error Detection Operation

In this reference design, initialization errors, writing data errors and reading data errors can be detected during transmission. If an error is detected in processing, the SD card will generate an error indication to the host. The host can read the TRANS_ERROR_REG register to judge the state of the controller.

The following steps outline the fundamental operation of the SD card.

Initialize

```
Set SPI_TRANS_TYPE_REG  = SPI_INIT_SD
Set SPI_TRANS_CTRL_REG  = SPI_TRANS_START
Wait for SPI_TRANS_STS_REG  != TRANS_BUSY
Check for SPI_TRANS_ERROR_REG [1:0] == INIT_NO_ERROR
```

Block Write

Write 512 bytes to SPI_TX_FIFO_DATA_REG
Set the SD block address registers:
SD_ADDR_7_0_REG
SD_ADDR_15_8_REG
SD_ADDR_23_16_REG
SD_ADDR_31_24_REG
Set SPI_TRANS_TYPE_REG = SPI_RW_READ_SD_BLOCK
Set SPI_TRANS_CTRL_REG = SPI_TRANS_START
Wait for SPI_TRANS_STS_REG! = TRANS_BUSY
Check for SPI_TRANS_ERROR_REG [5:4] == WRITE_NO_ERROR

Block Read

Set the SD block address registers:
SD_ADDR_7_0_REG
SD_ADDR_15_8_REG
SD_ADDR_23_16_REG
SD_ADDR_31_24_REG
Set SPI_TRANS_TYPE_REG = SPI_RW_READ_SD_BLOCK
Set SPI_TRANS_CTRL_REG = SPI_TRANS_START
Wait for SPI_TRANS_STS_REG! = TRANS_BUSY
Check for SPI_TRANS_ERROR_REG [3:2] == READ_NO_ERROR
Read 512 bytes from SPI_RX_FIFO_DATA_REG

Register Descriptions

Table 4 lists the registers used in this reference design. All registers are 8 bits wide.

Table 4. User Register

Register Name	Address	Type
SPI_MARSTER_VERSION_REG	0x0	Read
SPI_MARSTER_CONTROL_REG	0x1	Read/Write
TRANS_TYPE_REG	0x2	Read/Write
TRANS_CTRL_REG	0x3	Read/Write
TRANS_STS_REG	0x4	Read
TRANS_ERROR_REG	0x5	Read
DIRECT_ACCESS_DATA_REG	0x6	Read/Write
SD_ADDR_7_0_REG	0x7	Read/Write
SD_ADDR_15_8_REG	0x8	Read/Write
SD_ADDR_23_16_REG	0x9	Read/Write
SD_ADDR_31_24_REG	0xa	Read/Write
SPI_CLK_DEL_REG	0xb	Read/Write
RX_FIFO_DATA_REG	0x10	Read/Write
RX_FIFO_DATA_COUNT_MSB	0x12	Read/Write
RX_FIFO_DATA_COUNT_LSB	0x13	Read/Write
RX_FIFO_CONTROL_REG	0x14	Read/Write
TX_FIFO_DATA_REG	0x20	Read/Write
TX_FIFO_CONTROL_REG	0x24	Read/Write

Table 5 describes each register in detail. R denotes read only, W denotes write only and R/W denotes read and write.

Table 5. Register Detail

Register	Bit Position	Name	Description	Default	R/W
SPI_MARSTER_VERSION_REG	[7:4]	VERSION_NUM_MAJOR	Major revision number.	F	R
	[3:0]	VERSION_NUM_MINOR	Minor revision number.	F	R
SPI_MARSTER_CONTROL_REG	0	RST	1: Reset the core logic and registers. Self clearing.	0	W

TRANS_TYPE_REG	[1:0]	TRANS_TYPE	Sets the transaction type, where:
			2'b00:Direct Access;
			2'b01:INIT_SD;
			2'b10:RW_READ_SD_BLOCK;
			2'b11:RW_WRITE_SD_BLOCK;
				0	W

TRANS_CTRL_REG	0	TRANS_START	1'b1:Start transaction. Self clearing.	0	W
TRANS_STS_REG	0	TRANS_BUSY	1'b1:Transaction busy	0	R

TRANS_ERROR_REG	[5:4]	SD_WRITE_ERROR	2'b00:Write NO ERROR;
			2'b01:Write CMD ERROR;
			2'b10:Write DATA ERROR;
			2'b11:Write BUSY ERROR;
				0	R

	[3:2]	SD_READ_ERROR	2'b00:Read NO ERROR;
			2'b01:Read CMD ERROR;
			2'b10:Read TOKEN ERROR;
				0	R

	[1:0]	SD_INIT_ERROR	2'b00:INIT NO ERROR;
			2'b01:INIT CMD0 ERROR;
			2'b10:INIT CMD1 ERROR
				0	R

DIRECT_ACCESS_DATA_REG	[7:0]	TX_DATA	Set TX_DATA prior to starting a DIRECT-_ACCESS transaction. Note that the SPI bus has no concept of a read or write transaction. Thus, every DIRECT_ACCESS transaction transmits data from the SPI master and receives data from the SPI slave.	0	W
	[7:0]	RX_DATA	Read RX_DATA after completing a DIRECT-_ACCESS transaction.	0	W
SD_ADDR_7_0_REG	[7:0]	SD_ADDR_7_0	SD_ADDR [7:0].Normally set to zero because memory accesses should occur on a 512-byte boundary. Set the SD/MMC memory address before starting a block read or block write	00	R/W
SD_ADDR_15_8_REG	[7:0]	SD_ADDR_15_8	SD_ADDR [15:8]. Normally set SD_ADDR[8] to zero because memory accesses should occur on a 512-byte boundary	00	R/W
SD_ADDR_23_16_REG	[7:0]	SD_ADDR_23_16	SD_ADDR[23:16]	00	R/W
SD_ADDR_31_24_REG	[7:0]	SD_ADDR_31_24	SD_ADDR_31_24	00	R/W
SPI_CLK_DEL_REG	[7:0]	SPI_CLK_DEL	SPI_CLK_DEL controls the frequency of the SPI_CLK after SD initialization is complete. When setting clock frequency during SD initialization, you will need to modify the constant in spiMaster _defines.v SPI_CLK_DEL= (spiSysClk / (SPI_CLK * 2)) -1.	00	R/W
RX_FIFO_DATA_REG	[7:0]	RX_FIFO_DATA	SD/MMC block read data.	00	R
RX_FIFO_DATA_COUNT_MSB	[7:0]	FIFO_DATA_COUNT_MSB	MSByte of FIFO_DATA_COUNT. Indicates the number of data entries within the fifo.	00	R
RX_FIFO_DATA_COUNT_LSB	[7:0]	FIFO_DATA_COUNT_LSB	LSByte indicates the data entries within the FIFO.	00	R
RX_FIFO_CONTROL_REG	0	FIFO_FORCE_EMPTY	1 = Force FIFO empty. Deletes all the data samples within the FIFO. Self clearing.	00	W
TX_FIFO_DATA_REG	[7:0]	TX_FIFO_DATA	SD/MMC block write data. FIFO size matches the SD/MMC block size of 512 bytes.	00	W
TX_FIFO_CONTROL_REG	0	FIFO_FORCE_EMPTY	1 = Force FIFO empty. Deletes all the data samples within the FIFO. Self clearing.	00	W

Design Module Descriptions

The SD Flash Controller is the link between the microprocessor and the SD Flash SPI bus. It is responsible for timing and protocol between microprocessor accession and the SD Flash SPI bus. It consists of control and status registers, one 8-bit receive data FIFO which is 512 entries deep, and one 8-bit transmit FIFO which is 512 entries deep. The microprocessor reads and writes the SD Flash Controller registers and FIFOs in order to initiate communication to a card. The communication protocol of this reference design is based on SPI mode. The SD Flash Controller begins operation and reads the state of the SD card by accessing the user registers.

SpiMasterWishBoneBI Module

This module provides a bridge between the WISHBONE bus and the controller. The host can configure different user registers of the Controller to implement different operations. This module includes three user registers, the CTRL_STS_REG_BASE register, the RX_FIFO_BASE register and the TX_FIFO_BASE register. The CTRL_STS_REG_BASE register indicates the receiving data from the state user register, the RX_FIFO_BASE register indicates the receiving data from the receive FIFO, and the TX_FIFO_BASE indicates the receiving data from the transmit FIFO. At the same time, it generates an ack signal to the WISHBONE bus.

CtrlStsRegBI Module

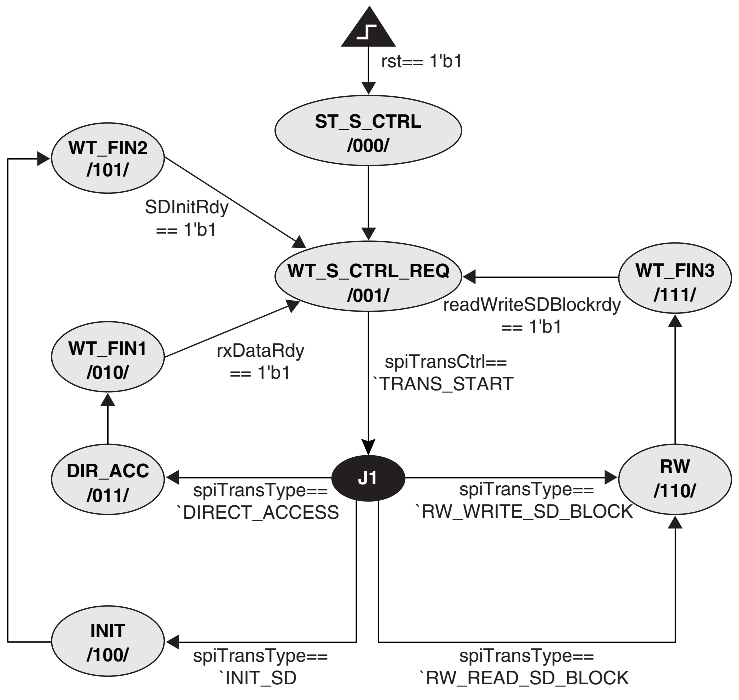
This module is the transfer control unit and includes four functions of the controller. According to the host configurating user register, it can generate the start signal of each operation, generate a direct accessing control signal, judge the type of the current configuration, monitor the internal state of the controller, configure the sector address of the current read or write operation, and implement the reset signal synchronous with the spiSysClk.

SpiCtrl Module

This module carries out the required operations based on the transfer type. It first waits the start signal of the operation, then judges the input transfer type to generate the reset request signal, initialization request signal, reading request signal and writing request signal.

Figure 5 shows the SpiCtrl module control state machine.

Figure 5. SpiCtrl Module State Machine



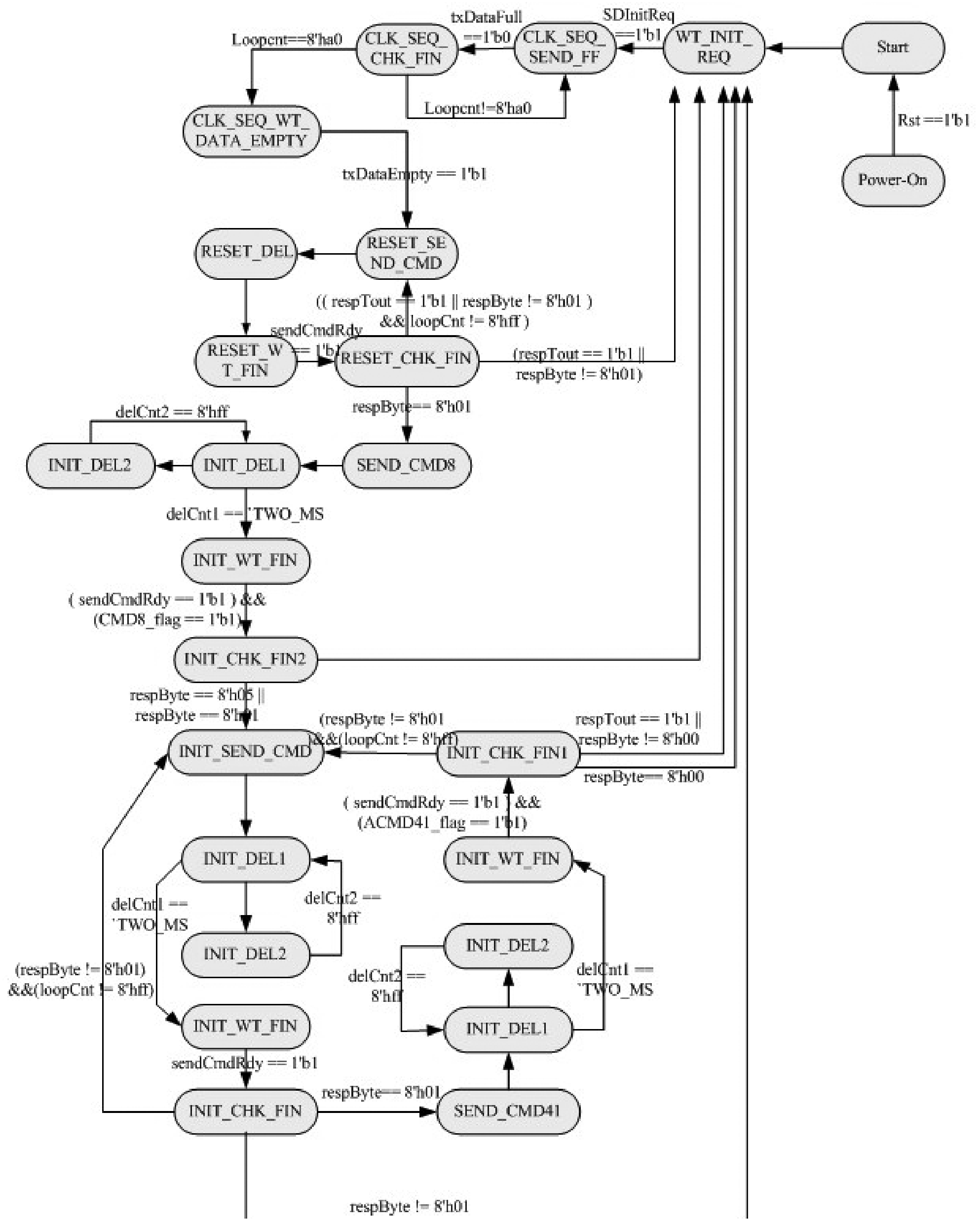
InitSD Module

This module implements the initialization function. When an SD card is powered up and the communication mode is SPI mode, the SD card enters the initialization process. The host must first generate the CMD0, CMD8, CMD55 and ACMD41 commands to start the initialization process to reset and initiate the SD card. For the simple implementation and saving resource, the CMD0 and CRC are implemented as parameters in this design.

The generated command byte is transferred to the send CMD module.

Figure 6 shows the main control state machine.

Figure 6. SD Card Initialization State Machine



ReadWriteSDBlock Module

This module implements data processing operations that read and write data from and into the SD card. In this reference design, the width of the address is 32 bits and the capacity is up to 2G bits. When the host wants to start reading or writing the SD card, it must configure the address of the operation. When a reading operation request is received from the microprocessor, the main state machine generates a read command (CMD17) to start the reading process. When a writing operation request is received, the main state machine generates a write command (CMD24) to start the writing operation. In this reference design, the Controller can only support single block operation.

The CMD17 and CRC are defined as follows:

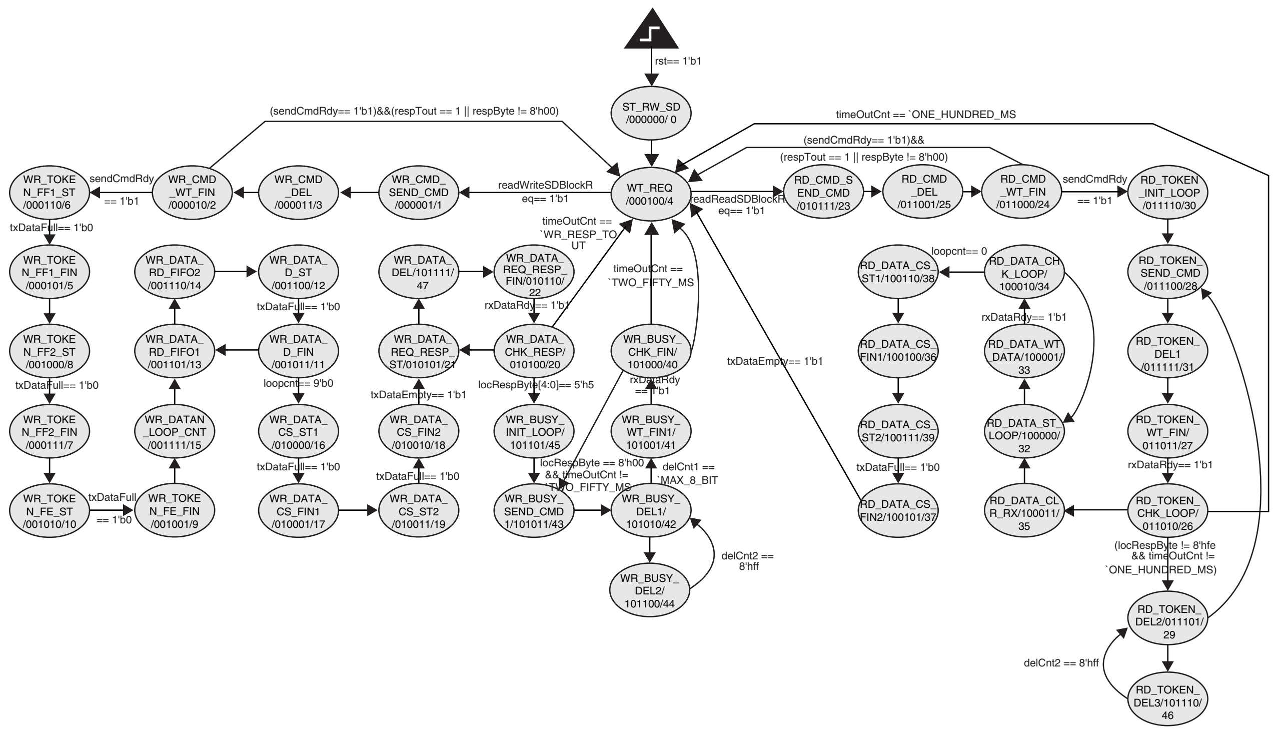
CMD_BYTE = 8’h51
Data_BYTE1=blockAddr[31:24]
Data_BYTE2= blockAddr[23:16]
Data_BYTE3= blockAddr[15:8]
Data_BYTE4= blockAddr[7:0]
CheckSumByte=8’hff

The CMD24 and CRC results (constant value 7’h3f) are defined as follows:

CMD_BYTE= 8’h58
Data_BYTE1=blockAddr[31:24]
Data_BYTE2= blockAddr[23:16]
Data_BYTE3= blockAddr[15:8]
Data_BYTE4= blockAddr[7:0]
CheckSumByte=8’hff

Figure 7 shows the read and write SD block state machine.

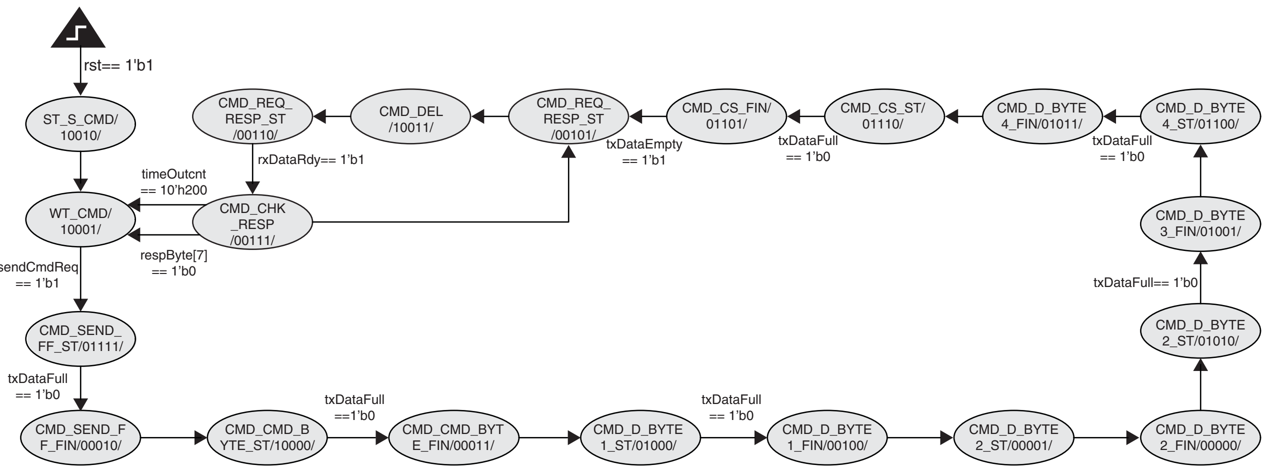
Figure 7. SD Card Read and Write SD Block State Machine


rst== 1'b1
ST_RW_SD /000000/ 0
WT_REQ /001000/4
(sendCmdRdy== 1'b1)&&(respTout == 1 || respByte != 8'h00)
timeOutCnt == `ONE_HUNDRED_MS
(sendCmdRdy== 1'b1)&&
(respTout == 1 || respByte != 8'h00)
readWriteSDBlockR eq== 1'b1
readReadSDBlockR eq== 1'b1
WR_TOKEN_FF1_ST /000110/6
sendCmdRdy == 1'b1
WR_CMD _WT_FIN /000010/2
WR_CMD _DEL /000011/3
WR_CMD_ SEND_CMD /000001/1
RD_CMD_S END_CMD /010111/23
RD_CMD _DEL /011001/25
RD_CMD _WT_FIN /011000/24
sendCmdRdy == 1'b1
RD_TOKEN _INIT_LOOP /011110/30
txDataFull== 1'b0
WR_TOKEN_FF1_FIN /000101/5
WR_DATA_ RD_FIFO2 /001110/14
WR_DATA_ D_ST /001100/12
WR_DATA DEL/101111/ 47
WR_DATA_ REQ_RESP_ FIN/010110/ 22
timeOutCnt == `WR_RESP_TO UT
timeOutCnt == `TWO_FIFTY_MS
loopcnt== 0
RD_DATA_CS_ ST1/100110/38
RD_DATA_CH K_LOOP/ 100010/34
RD_TOKEN_ SEND_CMD /011100/28
txDataFull== 1'b0
rxDataRdy== 1'b1
rxDataRdy== 1'b1
WR_TOKEN_FF2_ST /001000/8
WR_DATA_ RD_FIFO1 /001101/13
WR_DATA_ D_FIN /001011/11
WR_DATA_ REQ_RESP_ ST/010101/21
WR_DATA_ CHK_RESP/ 010100/20
WR_BUSY_ CHK_FIN/ 101000/40
txDataEmpty== 1'b1
RD_DATA_CS_ FIN1/100100/36
RD_DATA_WT _DATA/100001/ 33
RD_TOKEN_ DEL1 /011111/31
txDataFull== 1'b0
loopcnt== 9'b0
txDataEmpty== 1'b1
locRespByte[4:0]== 5'h5
rxDataRdy == 1'b1
WR_TOKEN_FF2_FIN /000111/7
WR_DATAN _LOOP_CNT /001111/15
WR_DATA_ CS_ST1 /010000/16
WR_DATA_ CS_FIN2 /010010/18
WR_BUSY_ INIT_LOOP/ 101101/45
WR_BUSY_ WT_FIN1/ 101001/41
RD_DATA_CS_ ST2/100111/39
RD_DATA_ST_ LOOP/100000/ 32
RD_TOKEN_ WT_FIN/ 011011/27
txDataFull== 1'b0
txDataFull== 1'b0
locRespByte == 8'h00 && timeOutCnt != `TWO_FIFTY_MS
delCnt1 == `MAX_8_BIT
txDataFull== 1'b0
rxDataRdy== 1'b1
WR_TOKEN_FE_ST /001010/10
txDataFull == 1'b0
WR_TOKEN_FE_FIN /001001/9
WR_DATA_ CS_FIN1 /010001/17
WR_DATA_ CS_ST2 /010011/19
WR_BUSY_ SEND_CMD 1/101011/43
WR_BUSY_ DEL1/ 101010/42
RD_DATA_CS_ FIN2/100101/37
RD_DATA_CL R_RX/100011/ 35
RD_TOKEN_ CHK_LOOP/ 011010/26
delCnt2 == 8'hff
WR_BUSY_ DEL2/ 101100/44
(locRespByte != 8'hfe && timeOutCnt != `ONE_HUNDRED_MS)
RD_TOKEN_ DEL2/011101/ 29
delCnt2 == 8'hff
RD_TOKEN_ DEL3/101110/ 46

SendCmd Module

This module receives the input command from the initialization unit and readwrite SD Block unit, receives the response token from the SD card and generates the time out signal as the indication signal. According to the received signals, the state machine generates the corresponding handshake signals to the related modules. At the same time, it receives the response type from the SD card to understand the SD card status. Figure 8 shows the main control state machine.

Figure 8. Send Command Module State Machines



SpiTxRxData Module

This module receives from four different branch data and write-enable signals which are from the SpiCtrl module, Sendcmd module, Init module and RDWRSDBlock module. According the write enable signal, it transmits receiving data as output to the SD card's data and generates the txdatafull signal to trigger the writing process of the SD card.

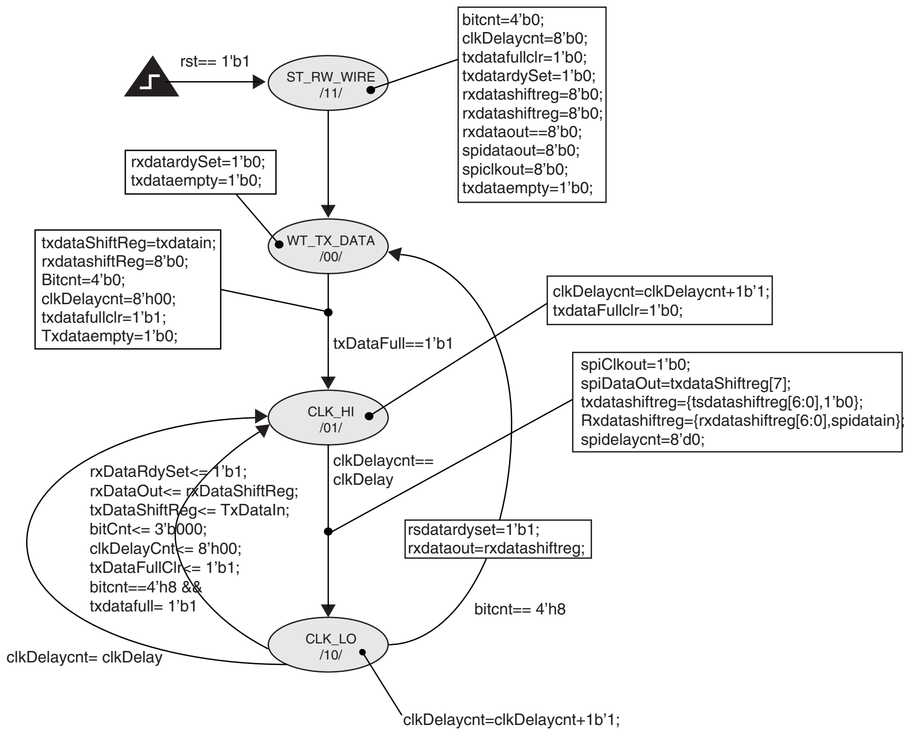
At the same time, it receives feedback data from the SD card. Based on the response signal, the module sends the response data to the host.

ReadWriteSPIWireData Module

This module receives command and data from the host and receives SD card data as the feedback data. It needs transfer receiving byte to serialize the data, then transmits and receives data to and from the SD card.

Figure 9 is the ReadWriteWireData module state machine.

Figure 9. ReadWriteSPIWireData State Machine



Sm_TxFifo Module

This module implements a simple FIFO and writes the host transmission data into Tx_fifo. It can force the Tx_fifo empty if the controller receive empty command with configuring the FIFO_CONTROL_REG register. It includes two modules, the sm_fifoRTL and the sm_TxfifoBI module.

Sm_Rxfifo Module

This module implements a simple FIFO and reads the SD transmission data into Rx_fifo. It can force the Rx_fifo empty if the transmission process generates an error when configuring the FIFO_CONTROL_REG register. It includes two modules, the sm_fifoRTL and the sm_RxfifoBI module.

Clock_switch Module

This is the control clock module for writing transmitted FIFO and reading received FIFO. When the user wishes to write data to the SD card, he must first switch the clock for writing transmitted FIFO, and then he can write the data to the transmitting FIFO and send the writing request signal to the Controller.

When the user wants to read the receiving data from the receiving FIFO, he must first switch the reading clock of the receiving FIFO, and then he can read the data from the receiving FIFO.

Test Bench Description

The test bench for this design includes the following modules:

- testHarness
- sdModel
- wb_master_model
- Operation flow

TestHarness Module

The testHarness module provides reset, busClk and spiSysClk to the user logic. The busClk is 25 MHz and the spiSysClk is 50 MHz.This module generates the test case to verify the function of the controller. The test case tests all of the user registers, writing a single block and reading a single block.

sdModel

The sdModel is a simple behavior model of the SD card. It receives the transmission data. This model needs the testHarness module to generate the respByte as the response byte. The output data from the SD card are all 8'hFF except the response byte.

wb_master_model

The wb_master_model module simply simulates the function of the microprocessor. It includes three tasks, writing a byte, reading a byte, and comparing the read data with the expected data.

Operation Flow

The basic unit of data transfer to/from the SD card is one byte. Any data transfer which requires a block size should define the block length as an integer in multiples of bytes.

The test case tests the following functions of this SD Flash Controller:

- Host direct access one physical address (one byte) with SPI mode
- Host reset and initialization SD card
- Host writes single block (512 bytes) to the SD card
- Host reads single block (512 bytes) from the SD card

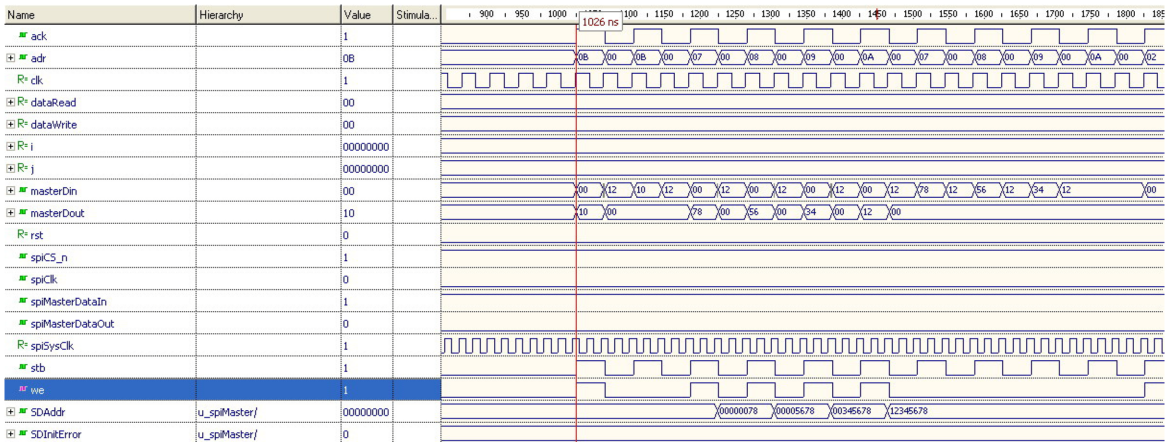
Timing Diagrams

Register Read and Write Test

This function tests the writing and reading of the user registers which include SPI_CLK_DEL_REG, SD_ADDR_7_0_REG, SD_ADDR_15_8_REG, SD_ADDR_23_16_REG and SD_ADDR_31_24_REG.

The host first writes 8'h10 to SPI_CLK_DEL_REG, and then writes the physical address register. The host writes the 8'h78 to address SD_ADDR_7_0_REG (8'h07), 8'h56 to address SD_ADDR_15_8_REG (8'h08), 8'h34 to address SD_ADDR_23_16_REG (8'h09), and 8'h12 to address SD_ADDR_31_24_REG (8'h0A).

Figure 10. Register Read and Write Test Timing Diagram

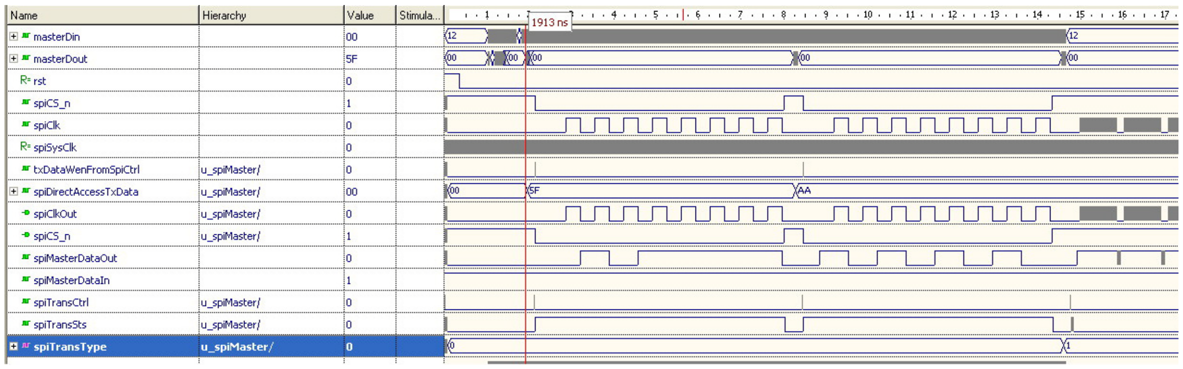


Name	Hierarchy	Value	Stimula...
ack		1	
adr		0B	
clk		1	
dataRead		00	
dataWrite		00	
i		00000000	
j		00000000	
masterDin		00	
masterDout		10	
rst		0	
spiCS_n		1	
spiClk		0	
spiMasterDataIn		1	
spiMasterDataOut		0	
spiSysClk		1	
stb		1	
we		1	
SDAddr	u_spiMaster/	00000000	
SDInitError	u_spiMaster/	0	

SPI Direct Accession

The host directly accesses the SD card and writes 8'h5f and 8'haa to the SD card. The user can access the responding user register.

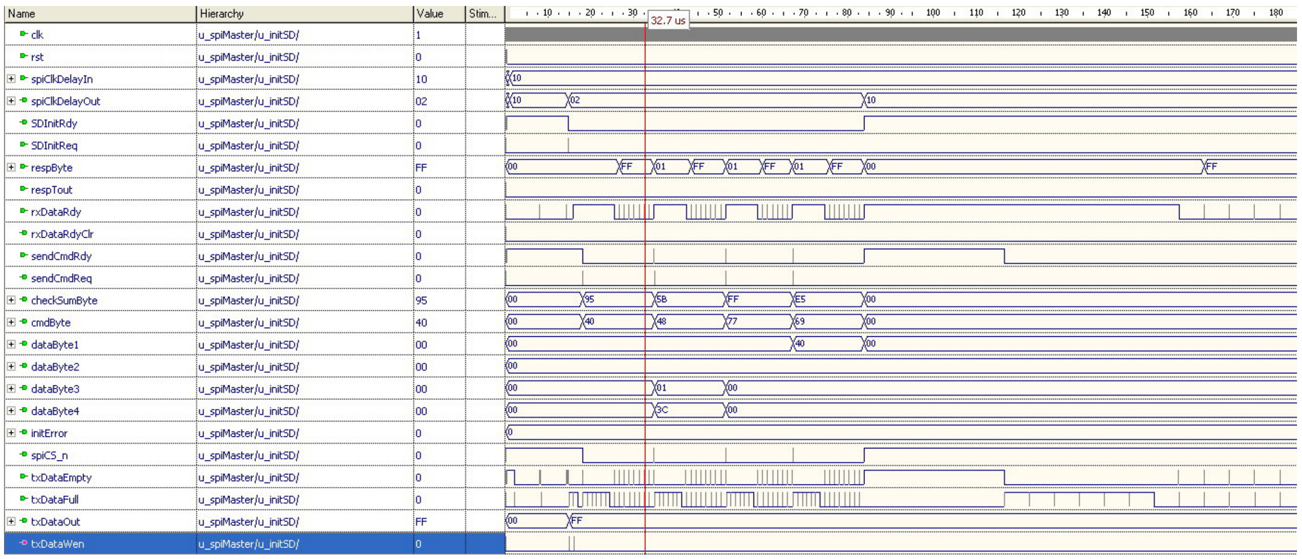
Figure 11. SPI Direct Accession Timing Diagram



Initialization

The host writes the TRANS_TYPE_REG and TRANS_CTRL_REG to start the initialization operation.The host sends CMD0, CMD8, CMD55 and ACMD41 to initialize the SD card.

Figure 12. Initialization Timing Diagram



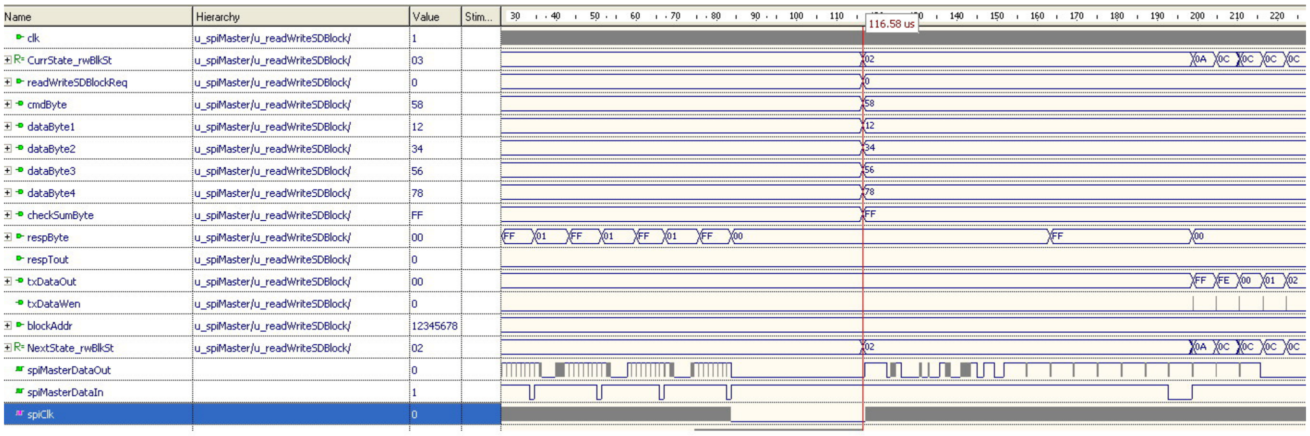
Writing SD Card

This operation requires the following steps.

1. Switch the clock to bus clock.
2. Empty the transmitting FIFO.
3. Write data to the transmitting FIFO.
4. Switch the clock to spiSysClk.
5. Toggle the transfer type.
6. Start to write the SD card.

The host sends data to the SD card after the CMD24 (8’h58).

Figure 13. Writing SD Card Timing Diagram



Name	Hierarchy	Value	Stim...
clk	u_spiMaster/u_readWriteSDBlock/	1	
CurrState_rwBlkSt	u_spiMaster/u_readWriteSDBlock/	03	
readWriteSDBlockReq	u_spiMaster/u_readWriteSDBlock/	0	
cmdByte	u_spiMaster/u_readWriteSDBlock/	58	
dataByte1	u_spiMaster/u_readWriteSDBlock/	12	
dataByte2	u_spiMaster/u_readWriteSDBlock/	34	
dataByte3	u_spiMaster/u_readWriteSDBlock/	56	
dataByte4	u_spiMaster/u_readWriteSDBlock/	78	
checkSumByte	u_spiMaster/u_readWriteSDBlock/	FF	
respByte	u_spiMaster/u_readWriteSDBlock/	00	
respTout	u_spiMaster/u_readWriteSDBlock/	0	
txDataOut	u_spiMaster/u_readWriteSDBlock/	00	
txDataWen	u_spiMaster/u_readWriteSDBlock/	0	
blockAddr	u_spiMaster/u_readWriteSDBlock/	12345678	
NextState_rwBlkSt	u_spiMaster/u_readWriteSDBlock/	02	
spiMasterDataOut		0	
spiMasterDataIn		1	
spiClk		0	

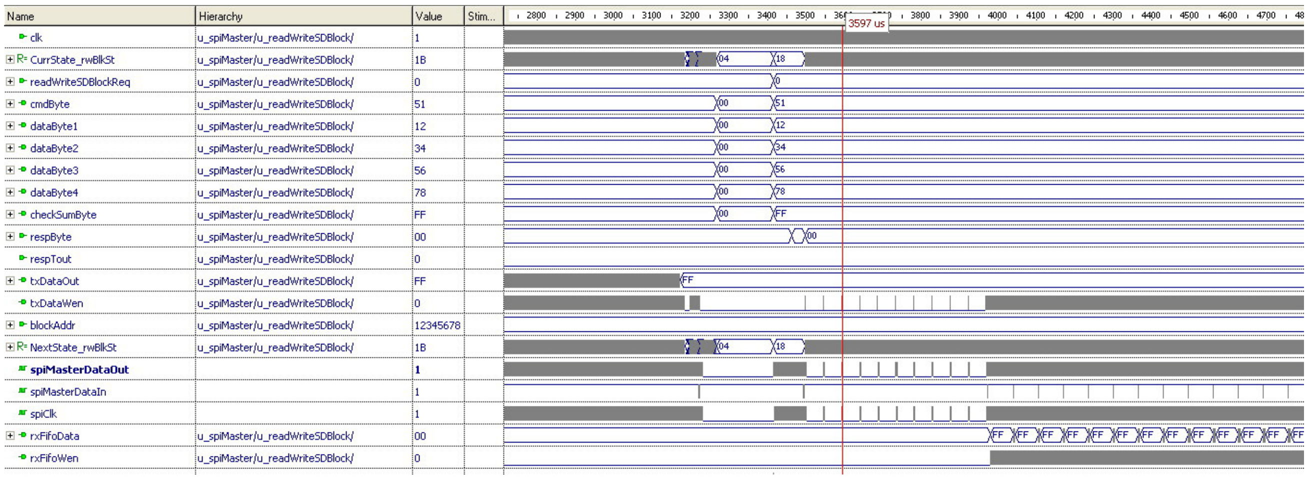
Reading SD Card

This operation requires the following steps.

1. Toggle the transfer type.
2. Start to write the SD card.
3. Judge whether the write receiving FIFO is complete or not.
4. Switch the reading clock to the bus clock.
5. Judge whether the read receiving FIFO is complete or not.
6. Switch the clock to spiSysClk.

The host reads data to the SD card after the CMD17 (8’h51).

Figure 14. Reading SD Card Timing Diagram



Name	Hierarchy	Value	Stim...
clk	u_spiMaster/u_readWriteSDBlock/	1	
CurrState_rwBlkSt	u_spiMaster/u_readWriteSDBlock/	1B	
readWriteSDBlockReq	u_spiMaster/u_readWriteSDBlock/	0	
cmdByte	u_spiMaster/u_readWriteSDBlock/	51	
dataByte1	u_spiMaster/u_readWriteSDBlock/	12	
dataByte2	u_spiMaster/u_readWriteSDBlock/	34	
dataByte3	u_spiMaster/u_readWriteSDBlock/	56	
dataByte4	u_spiMaster/u_readWriteSDBlock/	78	
checkSumByte	u_spiMaster/u_readWriteSDBlock/	FF	
respByte	u_spiMaster/u_readWriteSDBlock/	00	
respTout	u_spiMaster/u_readWriteSDBlock/	0	
txDataOut	u_spiMaster/u_readWriteSDBlock/	FF	
txDataWen	u_spiMaster/u_readWriteSDBlock/	0	
blockAddr	u_spiMaster/u_readWriteSDBlock/	12345678	
NextState_rwBlkSt	u_spiMaster/u_readWriteSDBlock/	1B	
spiMasterDataOut		**1**	
spiMasterDataIn		1	
spiClk		1	
rxFifoData	u_spiMaster/u_readWriteSDBlock/	00	
rxFifoWen	u_spiMaster/u_readWriteSDBlock/	0	

Implementation

Table 6. Performance and Resource Utilization

Device Family	Speed Grade	I/Os	f_{MAX} (MHz)	Utilization (LUTs)	Architecture Resources
MachXO™ [1]	-4	34	>50	909	1 EBR
LatticeXP2™ [2]	-5	34	>50	1088	1 EBR

1. Performance and utilization characteristics are generated using LCMXO2280C-4FT256C, with Lattice ispLEVER 8.0 software. When using this design in a different device, density, speed, or grade, performance and utilization may vary.
2. Performance and utilization characteristics are generated using LFXP2-5E-5TN144C, with Lattice ispLEVER 8.0 software. When using this design in a different device, density, speed, or grade, performance and utilization may vary.

References

- SD Physical Layer Simplified Specification 2.0 WISHBONE System-on-Chip Interconnection Architecture for Portable IP Cores
- SD Physical Layer Simplified Specification 2.0
- MMC/SD card controller (www.opencores.org/projects.cgi/web/spimaster/overview)

Technical Support Assistance

Hotline:	1-800-LATTICE (North America)
	+1-503-268-8001 (Outside North America)
e-mail:	techsupport@latticesemi.com
Internet:	www.latticesemi.com

Revision History

Date	Version	Change Summary
May 2009	01.0	Initial release.
January 2010	01.1	Added support for LatticeXP2 device family.
		Changed initialization code for each card.