Morphologic
Embedded FPGA Solutions
www.morphologic.dk

XPS SD Card Interface (v1.00a)

Introduction

This document defines the functional operation of the XPS SD Card Interface. The core allows interfacing with SD and MMC cards with interface with of up to 4 bits and up to 50 MHz interface clock speeds.

This document provides a description of the functional blocks, signal and configuration parameters of the cores. It also explains the registers available for programming the core as well as a set of procedures for initialisation, reading and writing to the SD/MMC card.

Features

- Supports PLBv4.6 bus
- Optional PLB bus master DMA function for high speed transfer and CPU offloading
- Autonomous SD command execution with CRC generation and checking
- Data channel supports 1 or 4 bit interfaces and hardware CRC generation and checking
- Optional high performance DMA channel using PLB master port
- Programmable SD clock rates including odd dividers
- SD clock stopping for power reduction
- GPIO bits for card detection and write protection detection
- Supports SDHC cards

Core short specs.	
Supported Device Family	Spartan-3 Spartan-6 Virtex-4 Virtex-5 Virtex-6
Core Version	V1_00_a
Resource utilization	
Slices	
Block RAMs	
Special Features	none
Core deliveries	
Documentation	Data Sheet
Design Files	VHDL & Netlist
Constraints File	N/A
Verification	On request
Instantiation Template	N/A
Ref. Design & Application Notes	N/A
Design Tool Requirements	
Xilinx Implementation Tools	ISE® 11.2 or later
Verification	Xilinx BFL tool set
Simulation	Modelsim SE/PE 6.5 or later
Synthesis	XST
Support	
Provided by Morphologic ApS	

1. Functional Description

1.1 Overview

The SD card interface core provides an interface between the relatively complex SD card protocol and the PLB bus/host controller. The controller simplify host operations by providing the following basic functions:

- State machine controlled command transmit and response sequencing
- Command CRC generation/checking in hardware
- Automatic data sector read and write operations with programmable sector count
- Data CRC generation/checking in hardware
- 1 to 8 bit serialisation/de-serialisation for command interface
- 1 / 4 bit to 32-bit data mux / demuxing for data interface
- Automatic data pausing when writing/erasing sectors
- Optional master PLB bus DMA controller for autonomous high speed data transfers
- GPIO bits for card detection, write protection state and card power on/off control.
- Programmable SD clock rate, from PLB bus clock/2 to PLB bus clock/255
- Interrupt generation for SD command completion
- Interrupt generation for SD data transfer completion (only for DMA interface)

Below is shown a block diagram of the SD interface IP core together with it's interface signals. The major functional blocks are described in the subsequent sections:

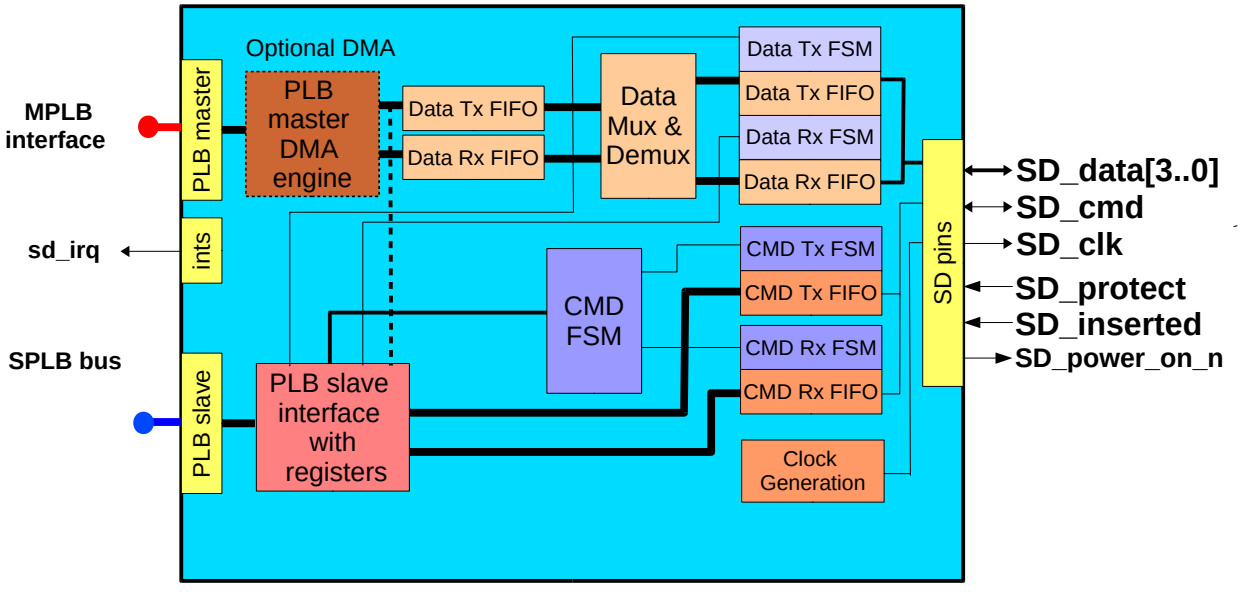

Fig 1. XPS_SD_IF IP block diagram.

1.2 PLB Master DMA engine (optional)

This interface is optional and can be disabled by assigning zero to the C_USE_DMA generic parameter. If disabled a 32-bit FIFO will be available for read/writes. If enabled, a DMA controller is added which can transfer single or multiple sectors to/from the SD card. The DMA controller can be configured to issue an interrupt when the transfer has been completed or an error occurred. The MPLB master uses a native 32-bit bus format, but will attach to 32, 64 or 128 bit buses.

1.3 PLB Slave Interface

The PLB slave uses a native 32-bit bus format, but will attach to 32, 64 or 128 bit buses. A block of 8 configuration registers are available through the interface. To simplify logic, the PLB slave only supports 32-bit read and writes. The PLB slave must use the same clock as the bus master port.

1.4 Command Transaction State Machine

The command transaction state machine controls the transmission and reception of responses on the 1-bit SD command interface. The state machine is controlled by the SD_CMD_CTRL register. When enabled by setting the SD_CMD_RUN bit in the SD_CMD_CTRL register, the data written to the 16 byte deep command Tx FIFO will be sent out on the command line until 5 bytes have been sent. The command bytes are followed by an automatically calculated 7-bit CRC and an end bit. The first byte sent must have bit-7=0 and bit-6=1 as per the SD card standard. The expected command response length is programmed into the SD_CMD_CTRL register. A response length of zero is allowed if no response is expected. When a response is received, the received bytes are written into the 16-byte deep command receive FIFO. The 7-bit CRC is not written to the FIFO but is compared to the CRC calculated from the received data. For some command responses the data used for the CRC calculation does not include the first byte received. In that case the CMD_CTRL_BIT_SKIP_BIT bit in the SD_CMD_CTRL register can be set to ignore and discard the first byte received. When the command response has been received (if any) an interrupt is asserted (if enabled) to notify the host that the received response is in the command Rx FIFO. Note that the SD clock is not stopped when the command Rx FIFO is full. If the received response is longer than 16 bytes (17 with the CMD_CTRL_BIT_SKIP_BIT bit set), the level in the Rx command FIFO must be kept below 16 bytes to avoid FIFO overrun.

When the SD_CMD_RUN bit in the SD_CMD_CTRL register is cleared the command Rx and Tx FIFOs are cleared.

1.4.1 Command Tx State Machine

The command Tx state machine handles the transmission of commands, command arguments and CRC-7 bits. If there are no bytes in the Tx FIFO and a command transaction is enabled, the SD clock will be stopped until there is data in the FIFO. The Tx state machine will always transmit 5 bytes (40 bits) followed by the CRC-7 and the end bit.

1.4.2 Command Rx State Machine

The command Rx state machine handles the reception of the command response bits. The state machine will wait for a start bit and will then receive the response bit and write them byte-wise to the Rx FIFO. The CRC-7 of the received bits will be calculated and then compared to the CRC-7 received at the end of the response. The result is indicated in the CMD_RX_CRC_BIT bit of the SD_CMD_STATUS register.

1.5 Data Tx and Rx State Machines

The data Tx and Rx state machines control the transmission and reception of data blocks. The interface can be set to use 1 or 4 data bits, according to the DATA_CTRL_BIT_4BIT_MODE_BIT of the DATA_CTRL register. In case of a 4-bit interface multiple CRCs are received/generated. The number of bytes per block for both Tx and RX is programmed in the SD_DATA_CTRL.

1.5.1 Data Tx State Machine

The transmit state machine when enabled will wait for data to be available in the transmit FIFO. Data can be provided by writing to the SD_DATA register in case of no DMA or it can be provided by the DMA engine when available. The Transmit FIFO is 16x32 bit deep. When data is available the SD clock will be enabled and data will be shifted out to the SD card starting with the start bit followed by data, CRC and the end bit. The SD receive status bit will then be received and should be read as “010” when there were no errors in the data received by the SD-card. If the DMA channel exists, transmission will stop if an error status is received. If the card becomes busy after reception of the data block, the state machine will wait for the busy condition to end before proceeding with the transmission of further blocks.

1.5.2 Data Rx State Machine

The receive state machine when enabled will enable the SD clock and then look for a start bit followed by the reception of a data block and CRC-16. The CRC-16 is not written to the Rx FIFO but is compared to the CRC-16 calculated on the received data. The status of the comparison is indicated in the DAT_CRC_STATUS_BIT bits of the SD_STATUS register (in case of a 1-bit interface only the least significant bit is valid). If DMA exists, the DMA will abort reception if a CRC error occurs. Should the receive FIFO which is 16 words deep become full, the SD clock is stopped until there is room in the FIFO again.

1.6 Data Mux and Demux

The data multiplexor and demultiplexer handles the change to/from 32-bit words from/to 4 or 1 bit units as required/provided by the SD card.

1.7 Data FIFOs

The data FIFOs are 32-bit wide and 16 word deep. The empty and full status bits of the Rx and Tx FIFOs are readable in the SD_STATUS register. The FIFOs are synchronous FIFO running of the PLB bus clock.

1.8 Clock generation

The clock for the SD core is generated by dividing the PLB bus by a programmable divider value and using a DDR output register at the SD clock output pin to generate the clock. The divider value can be anything from 2 to 1023. In case of odd values the output clock waveform is still a 50% duty cycle waveform. The clock is stopped when both the command and data interfaces are disabled or idle or when the Tx FIFO is full or the Rx FIFO is empty.

2. Core Interface Signals

2.1 Xilinx v4.6 PLB Slave Bus Signals

Signal Name	*Direction*	*Type*	*Description*
SPLB_Clk	I	std_logic	PLB main bus clock
SPLB_Rst	I	std_logic	PLB main bus reset
PLB_ABus	I	std_logic_vector(0 to 31)	PLB address bus
PLB_UABus	I	std_logic_vector(0 to 31)	PLB upper address bus
PLB_PAValid	I	std_logic	PLB primary address valid indicator
PLB_SAValid	I	std_logic	PLB secondary address valid indicator

PLB_rdPrim	I	std_logic	PLB secondary to primary read request indicator
PLB_wrPrim	I	std_logic	PLB secondary to primary write request indicator
PLB_masterID	I	std_logic_vector(0 to C_SPLB_MID_WIDTH-1)	PLB current master identifier
PLB_abort	I	std_logic	PLB abort request indicator
PLB_busLock	I	std_logic	PLB bus lock
PLB_RNW	I	std_logic	PLB read/not write
PLB_BE	I	std_logic_vector(0 to C_SPLB_DWIDTH/8-1)	PLB byte enables
PLB_MSize	I	std_logic_vector(0 to 1)	PLB master data bus size
PLB_size	I	std_logic_vector(0 to 3)	PLB transfer size
PLB_type	I	std_logic_vector(0 to 2)	PLB transfer type
PLB_lockErr	I	std_logic	PLB lock error indicator
PLB_wrDBus	I	std_logic_vector(0 to C_SPLB_DWIDTH-1)	PLB write data bus
PLB_wrBurst	I	std_logic	PLB burst write transfer indicator
PLB_rdBurst	I	std_logic	PLB burst read transfer indicator
PLB_wrPendReq	I	std_logic	PLB write pending bus request indicator
PLB_rdPendReq	I	std_logic	PLB read pending bus request indicator
PLB_wrPendPri	I	std_logic_vector(0 to 1)	PLB write pending request priority
PLB_rdPendPri	I	std_logic_vector(0 to 1)	PLB read pending request priority
PLB_reqPri	I	std_logic_vector(0 to 1)	PLB current request priority
PLB_TAttribute	I	std_logic_vector(0 to 15)	PLB transfer attribute
Sl_addrAck	O	std_logic	Slave address acknowledge
Sl_SSize	O	std_logic_vector(0 to 1)	Slave data bus size
Sl_wait	O	std_logic	Slave wait indicator
Sl_rearbitrate	O	std_logic	Slave re-arbitrate bus indicator
Sl_wrDAck	O	std_logic	Slave write data acknowledge
Sl_wrComp	O	std_logic	Slave write transfer complete indicator
Sl_wrBTerm	O	std_logic	Slave terminate write burst transfer
Sl_rdDBus	O	std_logic_vector(0 to C_SPLB_DWIDTH-1)	Slave read data bus
Sl_rdWdAddr	O	std_logic_vector(0 to 3)	Slave read word address
Sl_rdDAck	O	std_logic	Slave read data acknowledge

Sl_rdComp	O	std_logic	Slave read transfer complete indicator
Sl_rdBTerm	O	std_logic	Slave terminate read burst transfer
Sl_MBusy	O	std_logic_vector(0 to C_SPLB_NUM_MASTERS -1)	Slave busy indicator
Sl_MWrErr	O	std_logic_vector(0 to C_SPLB_NUM_MASTERS -1)	Slave write error indicator
Sl_MRdErr	O	std_logic_vector(0 to C_SPLB_NUM_MASTERS -1)	Slave read error indicator
Sl_MIRQ	O	std_logic_vector(0 to C_SPLB_NUM_MASTERS -1)	Slave interrupt indicator

2.2 Xilinx v4.6 PLB Master Bus Signals

Signal Name	*Direction*	*Type*	*Description*
MPLB_Clk	I	std_logic	PLB main bus Clock
MPLB_Rst	I	std_logic	PLB main bus Reset
M_request	O	std_logic	Master request
M_priority	O	std_logic_vector(0 to 1)	Master request priority
M_busLock	O	std_logic	Master buslock
M_RNW	O	std_logic	Master read/nor write
M_BE	O	std_logic_vector(0 to C_MPLB_DWIDTH/8-1)	Master byte enables
M_MSize	O	std_logic_vector(0 to 1)	Master data bus size
M_size	O	std_logic_vector(0 to 3)	Master transfer size
M_type	O	std_logic_vector(0 to 2)	Master transfer type
M_TAttribute	O	std_logic_vector(0 to 15)	Master transfer attribute
M_lockErr	O	std_logic	Master lock error indicator
M_abort	O	std_logic	Master abort bus request indicator
M_UABus	O	std_logic_vector(0 to 31)	Master upper address bus (unused)
M_ABus	O	std_logic_vector(0 to 31)	Master address bus
M_wrDBus	O	std_logic_vector(0 to C_MPLB_DWIDTH-1)	Master write data bus
M_wrBurst	O	std_logic	Master burst write transfer indicator
M_rdBurst	O	std_logic	Master burst read transfer indicator

PLB_MAddrAck	I	std_logic	PLB reply to master for address acknowledge
PLB_MSSize	I	std_logic_vector(0 to 1)	PLB reply to master for slave data bus size
PLB_MRearbitrate	I	std_logic	PLB reply to master for bus re-arbitrate indicator
PLB_MTimeout	I	std_logic	PLB reply to master for bus time out indicator
PLB_MBusy	I	std_logic	PLB reply to master for slave busy indicator
PLB_MRdErr	I	std_logic	PLB reply to master for slave read error indicator
PLB_MWrErr	I	std_logic	PLB reply to master for slave write error indicator
PLB_MIRQ	I	std_logic	PLB reply to master for slave interrupt indicator
PLB_MRdDBus	I	std_logic_vector(0 to (C_MPLB_DWIDTH-1))	PLB reply to master for read data bus
PLB_MRdWdAddr	I	std_logic_vector(0 to 3)	PLB reply to master for read word address
PLB_MRdDAck	I	std_logic	PLB reply to master for read data acknowledge
PLB_MRdBTerm	I	std_logic	PLB reply to master for terminate read burst indicator
PLB_MWrDAck	I	std_logic	PLB reply to master for write data acknowledge
PLB_MWrBTerm	I	std_logic	PLB reply to master for terminate write burst indicator

System signals:

Signal Name	*Direction*	*Type*	*Description*
IP2INTC_Irpt	O	std_logic	Active high interrupt line.

2.3 SD interface signals

Pin name	*Direction*	*Type*	*Description*
SD_cmd	I/O	std_logic	Bidirectional command line.
SD_data	I/O	std_logic_vector(3 downto 0)	Bidirectional data lines. If only a 1 bit interface is used, connect bit-0 only.
SD_clk	O	std_logic	Clock output

Pin name	Direction	Type	Description
SD_protect	I	std_logic	Card protect status bit
SD_inserted	I	std_logic	Card inserted status bit
SD_power_on_n	O	std_logic	Card power on output bit.

3. Core Parameters

Several generics allow customisation of the IP core to suit the type of display being attached to the IP signals, the pixel bitmap format, as well as the back-light and contrast control circuitry if any. Using generics allows for a smaller footprint core, leaving more room for other functions in the FPGA.

XPS_SD_IF Device Design Parameters

Generic name	type	Default value	Description
C_INCLUDE_DMA	integer	1	0=do not include DMA function 1=include DMA function

System and PLB parameters.

Name	Type	Allowable Values	Default value	Description
C_BASEADDR	std_logic_vector	Lower 6 bits must be '0'.	None	Slave register base address.
C_HIGHADDR	std_logic_vector	Lower 6 bits must be '1'.	None	Slave register high address.
C_SPLB_AWIDTH	integer	32	32	PLB Address Bus Width
C_SPLB_DWIDTH	integer	32, 64, 128	32	PLB Data Bus Width
C_SPLB_NUM_MASTERS	integer	1-16	8	Number of PLB masters
C_SPLB_MID_WIDTH	integer	log2(C_SPLB_NUM_MASTERS) with a minimum value of 1	1	Width of the PLB_master ID vector
C_SPLB_NATIVE_DWIDTH	integer	32	32	Width of slave data bus
C_SPLB_P2P	integer	0 : Shared bus topology 1 : Point to Point topology	0	PLB Point-to-Point or shared bus topology

C_SPLB_SUPPORT_BURSTS	integer	0:SPLB does not support bursts 1:SPLB supports bursts	0	unused
C_SPLB_SMALLEST_ MASTER	integer	32, 64, 128	32	Width of the smallest master that will be interacting with this slave. Unused
C_SPLB_CLK_PERIOD_PS	Integer	Any positive value	10000	SPLB clock period in picoseconds. Unused
C_INCLUDE_DPHASE_ TIMER	Integer	0:do include dataphase timer 1:include dataphase timer	1	Select whether a data phase timer in the slave interface is included or not.
C_FAMILY	String	spartan3e, spartan3a, spartan6, virtex4, virtex5, virtex6	virtex5	XILINX FPGA Family
C_MPLB_AWIDTH	Integer	32	32	Width of master address bus
C_MPLB_DWIDTH	Integer	32, 64, 128	32	Width of master data bus
C_MPLB_NATIVE_DWIDTH	Integer	32	32	Native width of IP master bus
C_MPLB_P2P	Integer	0 : Shared bus topology 1 : Point to Point topology	0	Master bus topology
C_MPLB_SMALLEST_SLAVE	Integer	32,64,128	32	Width of the smallest slave attached to the master bus
C_MPLB_CLK_PERIOD_PS	Integer	Any positive value	10000	MPLB clock period in picoseconds

4. Programming model

The following sections describe the programming model of the IP core. First a register map is given followed by a detailed description of the individual registers. Note that all bits are normally active high, unless explicitly otherwise mentioned.

4.1 Register map

All of the registers listed below must be accessed as 32-bit wide registers. All fields marked as reserved are read as 0 and written bits are ignored.

Register Name	*Offset from base address (hex)*
SD_CMD_DATA	0x00
SD_CLK_DIV	0x04
SD_CMD_CTRL	0x08
SD_STATUS	0x0C
SD_DATA	0x10
SD_DATA_CTRL	0x14

Register Name	*Offset from base address (hex)*
DMA_BASE	0x18
DMA_LENGTH	0x1C
DMA_CTRL	0x20
DMA_STATUS	0x24

4.1.1 SD_CMD_DATA (offset 0x0)

Name	*Bit*	*Type*	*Reset value*	*Descripton*
SD_CMD_DATA_FIFO	[7:0]	R/W	X	This field interface to the 8-bit command read and write FIFOs
Reserved	[31:8]	R	0	Reserved

4.1.2 SD_CLK_DIV (offset 0x4)

Name	*Bit*	*Type*	*Reset value*	*Descripton*
SD_CLK_DIVIDER	[9:0]	R/W	0	10-bit SD clock divider value. Odd values will still result in 50% duty cycle outputs. Minimum value is 2
Reserved	[30:10]	R/W	0	Reserved
SD_CMD_CTRL_POWER_ON_BIT	31	R/W	0	When set to '1', the sd_power_on_n output will go low.

4.1.3 SD_CMD_CTRL (offset 0x8)

Name	*Bit*	*Type*	*Reset value*	*Descripton*
SD_CMD_CTRL_ENA_BIT	0	R/W	0	Enables the command transaction state machine. When cleared the state machine is forced to abort any transaction started.
SD_CMD_CTRL_SKIP_BIT	1	R/W	0	When set, causes the first byte received from the command line to be discarded and ignored in the CRC-7 calculation.
SD_CMD_CTRL_WRDY_BIT	2	R/W	0	When set, causes the command transaction state machine to wait for a not-busy condition on the SD CMD line (a high) at the end of a transaction.
SD_CMD_CTRL_IRQ_ENA_BIT	3	R/W	0	Enables command interrupts at the end of a command transaction.

Name	*Bit*	*Type*	*Reset value*	*Descripton*
Reserved	[15:4]	R/W	0	Reserved
SD_CMD_CTRL_RX_LENGTH	[23:16]	R/W	0	Contains the expected number of bytes to receive as part of a command transaction. Can be zero.
Reserved	[31:24]	R/W	0	Reserved

4.1.4 SD_STATUS (offset 0x0C)

Name	*Bit*	*Type*	*Reset value*	*Descripton*
CMD_IDLE_BIT	0	RO	1	Indicates the idle state of the command transaction state machine.
CMD_TX_IDLE_BIT	1	RO	1	Indicates the idle state of the command transmit state machine.
CMD_RX_IDLE_BIT	2	RO	1	Indicates the idle state of the command receive state machine.
CMD_RX_CRC_BIT	3	RO	0	Indicates the status of the CRC-7 calculation of the last received command. 1 = CRC error detected.
CMD_RX_EMPTY_BIT	4	RO	1	Indicates the empty status of the command Rx FIFO.
CMD_TX_FULL_BIT	5	RO	0	Indicates the full status of the command Tx FIFO.
DAT_TX_IDLE_BIT	6	RO	1	Indicates the idle state of the data Tx state machine.
DAT_RX_IDLE_BIT	7	RO	1	Indicates the idle state of the data Rx state machine.
DAT_RX_EMPTY_BIT	8	RO	1	Indicates the empty status of the data Rx FIFO.
DAT_TX_FULL_BIT	9	RO	0	Indicates the full status of the data Tx FIFO.
SD_INSERTED_BIT	10	RO	X	Indicates the level of the SD_INSERTED input port.
SD_PROTECT_BIT	11	RO	X	Indicates the level of the SD_PROTECT_BIT input port.
Reserved	[15:12]	RO	0000	Reserved
DAT_TX_STAT_BIT	[18:16]	RO	000	Indicates the status bits returned after a block write to the SD card. “010” means no error. “101” means a CRC error occurred. “111” means no response received.
CMD_IRQ_BIT	19	R/WC	0	Indicates the state of the command sequence state machine interrupt.

Name	*Bit*	*Type*	*Reset value*	*Descripton*
DAT_CRC_STATUS_BIT	[23:20]	RO	0000	Indicates the result of each data lines CRC check. In 1-bit mode, only bit-20 is valid. '0' means no error. '1' mean CRC error detected. The status is valid after reception of a data block.
DAT_RX_COUNT_BIT	[27:24]	RO	0000	Number of words in the data Rx FIFO.
DAT_TX_COUNT_BIT	[31:28]	RO	0000	Number of words in the data Tx FIFO.

4.1.5 SD_DATA (offset 0x10)

Name	*Bit*	*Type*	*Reset value*	*Descripton*
SD_DATA_DATA	[31.0]	R/W	0	This field interfaces to the 32-bit data Rx and Tx FIFOs. Writes to this register is ignored and reads return 0, when DMA exist.

4.1.6 SD_DATA_CTRL (offset 0x14)

Name	*Bit*	*Type*	*Reset value*	*Descripton*
DATA_CTRL_BIT_TX_ENA_BIT	0	R/W	0	Enables the data Tx state machine
DATA_CTRL_BIT_RX_ENA_BIT	1	R/W	0	Enables the data Rx state machine.
DATA_CTRL_BIT_RXFIFO_RST_BIT	2	R/W	0	Resets the data receive FIFO. Write '0' to allow data to enter the FIFO.
DATA_CTRL_BIT_4BIT_MODE_BIT	3	R/W	0	Enables the 4-bit data mode. Only change this bit when both the data Tx and Rx channels are idle.
Reserved	[15:4]	R/W	0	Reserved
DATA_CTRL_LENGTH_BIT	[26:16]	R/W	0	Number of data bytes to transmit or receive per data block. This register is sampled at the start of transmission or reception.
Reserved	[31:27]	R/W	0	Reserved

4.1.7 DMA_BASE (offset 0x18)

Name	*Bit*	*Type*	*Reset value*	*Descripton*
SD_DMA_BASE	[31:2]	R/W	0	Base address of DMA transfer. Bit [1:0] is ignored and read as zeros. This register is sampled at the start of the first DMA transfer.

4.1.8 DMA_LENGTH (offset 0x1C)

Name	*Bit*	*Type*	*Reset value*	*Descripton*
SD_DMA_LENGTH	[15:0]	R/W	0	Number of data blocks to transfer. This register is sampled at the start of the first DMA transfer.
Reserved	[31:16]	R/W	0	Reserved

4.1.9 DMA_CTRL (offset 0x20)

Name	*Bit*	*Type*	*Reset value*	*Descripton*
DMA_ENABLE_BIT	0	R/W	0	Enables DMA transfers. The DMA can be aborted after every access to the bus.
DMA_DIRECTION_BIT	1	R/W	0	'0' means write to PLB bus for SD data reads. '1' means read from PLB bus for SD data writes. This register is sampled at the start of the first DMA transfer.
DMA_IRQ_ENA_BIT	2	R/W	0	Enables DMA completion interrupts.
Reserved	[31:3]	R/W	0	Reserved

4.1.10 DMA_STATUS (offset 0x24)

Name	*Bit*	*Type*	*Reset value*	*Descripton*
SD_DMA_STAT_BLK_CNT	[15:0]	RO	0	Indicates the number of data blocks remaining to transfer by the DMA.
SD_DMA_STAT_IRQ_BIT	16	R/WO	0	Indicates the state of the DMA interrupt line.
SD_DMA_STAT_IDLE_BIT	17	R/W	1	Indicates the idle state of the DMA.
SD_DMA_STAT_TX_ERR_BIT	18	R/W	0	Indicates that a status code other

Name	Bit	Type	Reset value	Descripton
				than “010” was received from the SD card after a transmission of a data block to the SD. Cleared by disabling the transmitter. An error will abort the DMA transfer and generate an interrupt.
SD_DMA_STAT_RX_ERR_BIT	19	R/W	0	Indicates that a CRC error occurred when receiving a data block from the SD card. Cleared by disabling the receiver.
SD_DMA_STAT_BUS_ERR_BIT	20	R/W	0	Indicates that a PLB bus error occurred while doing a read or write from/to the PLB bus. Cleared by disabling the DMA.
Reserved	[31:21]	R/W	0	Reserved

5. Connecting the IP core to the SD card/connector.

The SD core is connected to the SD card using the signals listed in the table below. Auxiliary signals for power control, card detection and write protection sensing are also listed:

Core port name	SD pin name	SD pin number	Direction from core	Description
SD_data[0]	SD_DAT0	7	I/O	SD data 0 line for 1 or 4 bit mode
SD_data[1]	SD_DAT1	8	I/O	SD data 1 line for 4 bit mode
SD_data[2]	SD_DAT2	9	I/O	SD data 2 line for 4 bit mode
SD_data[3]	SD_DAT3	1	I/O	SD data 3 line for 4 bit mode
SD_clk	CLK	5	O	Clock line for SD commands and data.
SD_cmd	CMD	2	I/O	SD command/response data line
SD_protect	-	-	I	Originates from the SD card connector
SD_inserted	-	-	I	Originates from the SD card connector
SD_power_on_n	-	-	O	Normally connected to a P-type MOS transistor to turn on/off power to the SD card.

Pull-ups to the SD (P-MOS switched) power supply (pin 4) should be connected to DAT0, DAT1 and DAT2 (DAT3 has internal pull-up unless disabled by a SD command). SD_protect and SD_inserted normally also needs a pull-up or down, depending on the desired active level.

6. Programming the Core

This sections describes how to set up and control the XPS_SP_IF core so that SD card command

and data transactions can be performed.

6.1 Initialisation Sequence

After reset, the user only needs to configure the SD_CLK_DIV register before SD command and data requests can be executed. The clock divider divides the PLB bus clock frequency by an integer and outputs the clock on the SD_clk output pin. This output is generated using a double data rate output primitive. This allows the clock output stage to generate a 50% duty cycle output clock even in the case of odd dividers like 3 or 5, 7 etc. This in turn allows for a higher SD clock rate without the need for a separate clock and associated clock domain transfer logic. The SD_clk output frequency is given by:

$F_{SD} = F_{PLB} / N$

Where N is the value of the SD_CLK_DIVIDER field of the SD_CLK_DIV register.
F_{PLB} is the PLB bus frequency.
F_{SD} is the clock frequency of the SD_clk output.

Note that when performing initialisation of a SD card, the card should only run at a maximum of 400 KHz clock. After initialisation, a standard SD card supports clock speeds up to 25 MHz and a high speed card supports clock speeds up to 50 MHz (these max. values can also be read out from the SD card).

The SD clock line will be low when idle, to allow the SD card to be switched off if needed, without the clock line sourcing power to the SD card via it's internal protection diode.

6.2 Performing command transactions

Commands issued to a SD card always contain a 5 byte payload followed by a byte containing the CRC-7 and the end bit ('1'). The core will generate the CRC-7 and end bit. The first byte of a command should have bit-7=0 and bit-6=1 as these are not automatically forced by the core. The command response can have different lengths depending on the command issued. The expected command response length (excluding any CRC-7 and end bit) must be written to the SD_CMD_CTRL_RX_LENGTH field of the SD_CMD_CTRL register before issuing the command transaction, by setting the SD_CMD_CTRL_ENA_BIT bit of the same register to '1'. For some commands, no response is expected and it is legal in that case to program the response length to zero. After a response is received, the status of the CRC-7 check is reflected in the CMD_RX_CRC_BIT bit of the SD_STATUS register. Note that for some responses there is no CRC or response and the CMD_RX_CRC_BIT bit should be ignored. In summary the following steps should be performed to issue a command request and response sequence:

1. Set expected command response length in the SD_CMD_CTRL_RX_LENGTH field of the CMD_CTRL register.
2. Indicate in the SD_CMD_CTRL_SKIP_BIT bit whether the first byte of the response should be included in the response CRC calculation or not.
3. Indicate in the SD_CMD_CTRL_WRDY_BIT bit whether the core should wait for a ready condition after reception of the command response.
4. Indicate in the SD_CMD_CTRL_IRQ_ENA_BIT bit whether an interrupt should be generated after the transaction finishes.
5. Write the 5 SD command bytes to the command Tx FIFO register.
6. Set the SD_CMD_CTRL_ENA_BIT bit to start the response.
7. Enable any time-out timer needed to detect no-response situations.
8. Wait for CMD_IDLE_BIT to be set in the SD_STATUS register or for a timer time-out to

occur.

9. Check for a command response CRC error (if valid) in the SD_STATUS register.
10. Read out the command response byte (if any) from the command Rx FIFO at register SD_CMD_DATA.

Note that if a command response longer than 16 bytes is expected, the Rx FIFO must be emptied before the command transaction state machine can become idle. The CMD_RX_EMPTY_BIT of the SD_STATUS register can be checked to see if there is data in the Rx FIFO.

6.3 Performing SD data reads

SD card data blocks reads are normally initiated by issuing a command transaction, requesting the readout of data from the SD card starting from a specified data block. Some card status/identification registers are also read out as a single block of data. Upon completion of the command, the SD card will wait for clocks to appear, which will then read out the data.

One or more data blocks can be read at a time and can be transferred either via the optional DMA interface. Once the transaction using DMA is set up, the DMA can transfer the amount of data requested without further CPU intervention, freeing up CPU time for other tasks. An interrupt can be generated at the end of the DMA transfer or when an error occurs.

When the receive data state machine is enabled, the state machine will wait for the start bit and will then receive the number of bytes indicated in the DATA_CTRL_LENGTH_BIT field of the SD_DATA_CTRL register. The CRC-16 will then be received followed by the end bit. After reception of the CRC-16, the DAT_CRC_STATUS_BIT field will contain the result of the CRC check on the data lines used (1 or 4 bits). A CRC error on (any of) the data line(s) will cause the DMA to stop and generate an interrupt if enabled. The CRC status bit(s) will be cleared by clearing the DATA_CTRL_BIT_RX_ENA_BIT of the DATA_CTRL register.

If a bus error occurs on the PLB bus, the DMA will stop and the SD_DMA_STAT_BUS_ERR_BIT of the DMA_STATUS register will be set. The error condition can be cleared by disabling the DMA.

The data Rx FIFO can be cleared by briefly setting the DATA_CTRL_BIT_RXFIFO_RST_BIT to '1'. Reception of data can be aborted by clearing the DATA_CTRL_BIT_RX_ENA_BIT of the DATA_CTRL register. The Rx FIFO should then be cleared.

If the data Rx FIFO gets full, the SD clock will be stopped until there is room again. This also means that any command transactions are also halted as the data and command interfaces share the same clock line. The following operations should be performed after core initialisation, to receive data blocks using DMA:

1. Set the data block size in bytes, the SD data width in the DATA_CTRL_LENGTH_BIT field of the DATA_CTRL register.
2. Set the number of data lines that the SD card has been instructed to use by initialising the DATA_CTRL_BIT_4BIT_MODE_BIT bit of the DATA_CTRL register.
3. Toggle the DATA_CTRL_BIT_RXFIFO_RST_BIT in the DATA_CTRL register on and off to reset the Rx data FIFO (this step is optional).
4. Set the number of blocks to transfer in the SD_DMA_LENGTH field of the DMA_LENGTH register.
5. In the DMA_CTRL register, set the DMA_DIRECTION_BIT to '0', the DMA_IRQ_ENA_BIT to '1' (if an interrupt is wanted at the end of data block transfer) and finally set the DMA_ENABLE_BIT to '1' to enable DMA transfers.
6. Issue SD command transaction(s) to enable the SD card to return data block(s).
7. Set the DATA_CTRL_BIT_RX_ENA_BIT of the SD_DATA_CTRL register to enable the SD clock and to enable reception of data block(s).
8. Wait for an interrupt from the DMA engine. While waiting, the number of blocks remaining

to transfer can be read from the SD_DMA_STAT_BLK_CNT field.

9. Check the SD_DMA_STAT_IDLE_BIT, SD_DMA_STAT_RX_ERR_BIT and SD_DMA_STAT_BUS_ERR_BIT of the DMA_STATUS register and handle any errors.
10. Issue SD command transaction(s) to stop further transfer of SD data blocks.
11. Clear the DATA_CTRL_BIT_RX_ENA_BIT of the SD_DATA_CTRL register to disable further data transactions.
12. Clear the DMA_ENABLE_BIT of the DMA_CTRL register to disable the DMA engine.

The following operations should be performed after core initialisation, to receive data blocks using the polled interface:

1. Set the data block size in bytes, the SD data width in the DATA_CTRL_LENGTH_BIT field of the DATA_CTRL register.
2. Set the number of data lines that the SD card has been instructed to use by initialising the DATA_CTRL_BIT_4BIT_MODE_BIT bit of the DATA_CTRL register.
3. Toggle the DATA_CTRL_BIT_RXFIFO_RST_BIT in the DATA_CTRL register on and off to reset the Rx data FIFO (this step is optional).
4. Issue SD command transaction(s) to enable the SD card to return data block(s).
5. Set the DATA_CTRL_BIT_RX_ENA_BIT of the SD_DATA_CTRL register to enable the SD clock and to enable reception of data block(s).
6. Poll the received FIFO status by reading the DAT_RX_EMPTY_BIT and/or the DAT_RX_COUNT_BIT bits of the SD_STATUS register, followed one or more reads from the SD_DATA register. Read as much data as required.
7. Issue SD command transaction(s) to stop further transfer of SD data blocks.
8. Clear the DATA_CTRL_BIT_RX_ENA_BIT of the SD_DATA_CTRL register to disable further data transactions.
9. Toggle the DATA_CTRL_BIT_RXFIFO_RST_BIT in the DATA_CTRL register on and off to reset any remaining data in the the Rx data FIFO.

6.4 Performing SD data writes

SD card data blocks writes are normally initiated by issuing a command transaction, requesting the write of data by the SD card starting from a specified data block. Some card status/identification registers can also be written as a single block of data. Upon completion of the command, the SD card will wait for clock and data to appear, which it will the receive.

One or more data blocks can be read at a time and can be transferred either via the optional DMA interface. Once the transaction using DMA is set up, the DMA can transfer the amount of data requested without further CPU intervention, freeing up CPU time for other tasks. An interrupt can be generated at the end of the DMA transfer or when an error occurs.

When the transmit data state machine is enabled, the state machine will issue the start bit followed by the number of bytes indicated in the DATA_CTRL_LENGTH_BIT field of the SD_DATA_CTRL register. The CRC-16 is then calculated on the fly and is transmitted after the last byte of the data block followed by the end bit. The SD card will then return the status bits, indicating the CRC status of the data received. After reception of the status bits, the DAT_TX_STAT_BIT field of the SD_STATUS register will contain the status bits. Any error status will cause the DMA to stop and generate an interrupt if enabled. The CRC status bit(s) will be cleared by clearing the

DATA_CTRL_BIT_TX_ENA_BIT of the DATA_CTRL register.

If a bus error occurs on the PLB bus, the DMA will stop and the SD_DMA_STAT_BUS_ERR_BIT of the DMA_STATUS register will be set. The error condition can be cleared by disabling the DMA.

Transmission of data can be aborted by clearing the DATA_CTRL_BIT_TX_ENA_BIT of the DATA_CTRL register.

If the data Tx FIFO gets empty, the SD clock will be stopped until there is new data to transmit again. As a command transaction can re-start the clock, care must be taken not to issue a command until a whole data block has been transferred. The following operations should be performed after core initialisation, to transmit data blocks using DMA:

1. Set the data block size in bytes, the SD data width in the DATA_CTRL_LENGTH_BIT field of the DATA_CTRL register.
2. Set the number of data lines that the SD card has been instructed to use by initialising the DATA_CTRL_BIT_4BIT_MODE_BIT bit of the DATA_CTRL register.
3. Set the number of blocks to transfer in the SD_DMA_LENGTH field of the DMA_LENGTH register.
4. In the DMA_CTRL register, set the DMA_DIRECTION_BIT to '0', the DMA_IRQ_ENA_BIT to '1' (if an interrupt is wanted at the end of data block transfer) and finally set the DMA_ENABLE_BIT to '1' to enable DMA transfers.
5. Issue SD command transaction(s) to enable the SD card to receive data block(s).
6. Set the DATA_CTRL_BIT_TX_ENA_BIT of the SD_DATA_CTRL register to enable transmission of data block(s).
7. Wait for an interrupt from the DMA engine. While waiting, the number of blocks remaining to transfer can be read from the SD_DMA_STAT_BLK_CNT field.
8. Check the SD_DMA_STAT_IDLE_BIT, SD_DMA_STAT_TX_ERR_BIT and SD_DMA_STAT_BUS_ERR_BIT of the DMA_STATUS register. Handle any error conditions.
9. Issue SD command transaction(s) to stop further reception of SD data blocks.
10. Clear the DATA_CTRL_BIT_TX_ENA_BIT of the SD_DATA_CTRL register to disable further data transactions.
11. Clear the DMA_ENABLE_BIT of the DMA_CTRL register to disable the DMA engine.

The following operations should be performed after core initialisation, to receive data blocks using the polled interface:

1. Set the data block size in bytes, the SD data width in the DATA_CTRL_LENGTH_BIT field of the DATA_CTRL register.
2. Set the number of data lines that the SD card has been instructed to use by initialising the DATA_CTRL_BIT_4BIT_MODE_BIT bit of the DATA_CTRL register.
3. Issue SD command transaction(s) to enable the SD card to receive data block(s).
4. Set the DATA_CTRL_BIT_TX_ENA_BIT of the SD_DATA_CTRL register to enable transmission of data block(s).
5. Poll the received FIFO status by reading the DAT_TX_FULL_BIT and/or the DAT_TX_COUNT_BIT bits of the SD_STATUS register, followed one or more write to the SD_DATA register. Write as much data as required.
6. Issue SD command transaction(s) to stop let the SD card stop expecting further data.

7. Clear the DATA_CTRL_BIT_TX_ENA_BIT of the SD_DATA_CTRL register to disable further data transactions.

7. Resource Utilization

The core resources used in the spartan-6, virtex-5, and spartan-3 series are detailed in the table below:

Device family	Resource	With DMA	Without DMA
Spartan-6	Slice LUTs	1194	741
	Slice Registers	904	611
	BRAM18	0	0
	BRAM9	0	0
Virtex-5	Slice LUTs	1194	734
	Slice Registers	919	608
	BRAM36	0	0
	BRAM18	0	0
Spartan-3	Slices	807	522
	BRAM18	0	0
	BRAM9	0	0

8. Revision history

Date	Version	Revision
October 12th, 2011	1.0	Initial release

9. Notice of Disclaimer

Morpholgic ApS is providing this product documentation, hereinafter “Information,” to you “AS IS” with no warranty of any kind, express or implied. Morpholgic ApS makes no representation that the Information, or any particular implementation thereof, is free from any claims of infringement. You are responsible for obtaining any rights you may require for any implementation based on the Information. All specifications are subject to change without notice. MORPHOLOGIC APS EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE INFORMATION OR ANY IMPLEMENTATION BASED THEREON, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF INFRINGEMENT AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Except as stated herein, none of the Information may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Morpholgic ApS.