logiSDHC

User's Manual

Version: 1.07.a

logiSDHC_hum_v1.07.a.doc

1 INTRODUCTION

The logiSDHC – Secure Digital Host Controller is an IP core optimized for Xilinx FPGA's. It enables data transfers, between host system and SD card peripheral, through standardized interfaces according to Secure Digital Specification Version 2.00.
The adoptability to the different system specifications and requirements, at the same time optimizing the slice count utilization, is achieved through VHDL code parameterization.
Its functions include performing the DMA and non DMA data transfers as well as the non data transfers between the Host system and SD Card.
To achieve portability of the core source code across various Xilinx FPGA families, all IO blocks, clock drivers, and family-specific circuits are marked for easy replacement.
In order to simplify handling of logiSDHC features the software driver is made and included in the logiSDHC deliverables.
Apart from the above mentioned, there are a number of application notes, reference designs and evaluation boards available upon request. All these simplify integration of the logiSDHC with your design, shortening time to the market and increasing your market window.

1.1 GENERAL DESCRIPTION

The logiSDHC makes it easy to add SD Card peripheral to the embedded (HOST) system. It provides a "programmed I/O" method for the Host Driver to transfer data using the Buffer Data Port register. It also supports data transfer using DMA by implementing DMA algorithm defined in the SD Host Controller Standard Specification Version 1.00 also called SDMA (Single Operation DMA). This algorithm defines the data transfer's where only one SD command transaction is executed per DMA operation.
It is also possible to choose to implement additional (non standard) Xylon specific DMA controller. More information about that option is given in the chapter Xylon – specific DMA controller.

For easier system integration logiSDHC uses Xilinx (IBM) CoreConnect PLBv4.6 and OPB buses, AMBA AXI4 and a special Xylon Memory Bus (XMB). PLBv4.6, OPB and AXI4-Lite buses are available to choose for implementation of register interface. Xylon Memory Bus (XMB) and AMBA AXI4 are implemented as memory interface used for DMA transfers. Memory accesses for DMA transfers are designed in such a way that the required memory bandwidth is optimized and kept as low as possible.

1.2 FEATURES

- Supports Xilinx® Zynq™-7000 AP SoC and all Xilinx FPGA families
- Compliant with Secure Digital Specifications Version 2.00
- Supported both Standard and High Capacity SD cards
- Programmable transfer rates up to standard specific maximum rate of 25 MB/sec and 50 MHz bus frequency
- Supports non DMA, standard DMA and custom Xylon-specific DMA data transfers
- The standard DMA supporting enhanced features:
    - Read/write memory burst cycles (16,32 or 64 clock cycle bursts)
    - Interrupted memory burst cycles
    - Byte address boundary memory accesses
    - DMA interrupt mechanism according to the Standard
- Xylon-specific DMA supporting:
    - accelerated read (SD Card to Host) transfers of the bitmap files
    - Read memory burst cycles (16, 32 or 64 clock cycle bursts)
    - Interrupted memory burst cycles
    - Byte address boundary memory accesses
    - DMA interrupt generated when the transfer is completed
    - Supports endianness conversion
- Memory interface for DMA transfers (32 bits) is designed for Xylon logiMEM memory controller and other Xilinx memory controllers
- Buffer overflow detection functionality implemented to detect if the read transfer (from the SD Card to Host) is performed faster than the Host can handle
- Read wait functionality implemented to pause the read transfer (from the SD Card to Host), thus preventing possible buffer overflow error by giving the Host extra time to handle the data read from the Card
- Configurable register interface (32 bits) OPB, PLB or AXI4-Lite
- Simple Plug'n'Play with other Xylon logicBRICKSTM IP cores, such as:
    - logiMEM Flexible Memory Controller

1.3 REFERENCE DOCUMENTS

The logiSDHC User's Manual is based on and refers extensively on SD documentation:

- SD Specifications: SD Host Controller Specification Version 2.00
- SD Specifications: Physical Layer Specification Version 2.00

2 CORE ARCHITECTURE

The logiSDHC core consists of:

- Host Interface,
- Clock Generator,
- Buffer Controller,
- DMA Controller,
- SD Card Interface.

2.1 BLOCK SCHEMATIC

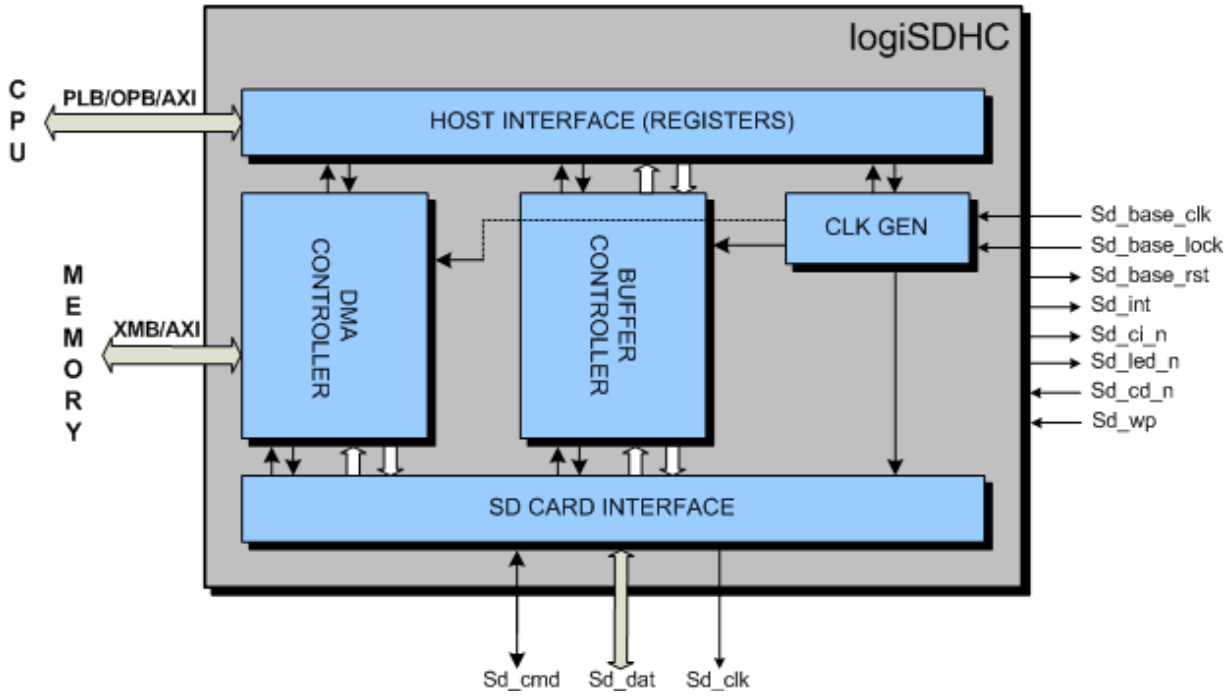

Figure 2.1: logiSDHC block schematic

2.1.1 Host Interface

The host interface module implements a layer through which the logiSDHC is connected to the host system. It contains standardized register set according to SD Specification Version 2.00 which will be described in chapter REGISTERS.

Functions of the logiSDHC are defined through the register settings, so this module also decodes register settings and generates command and control signals which run the functionality of logiSDHC. Classification of the standard register map is presented in a figure below:

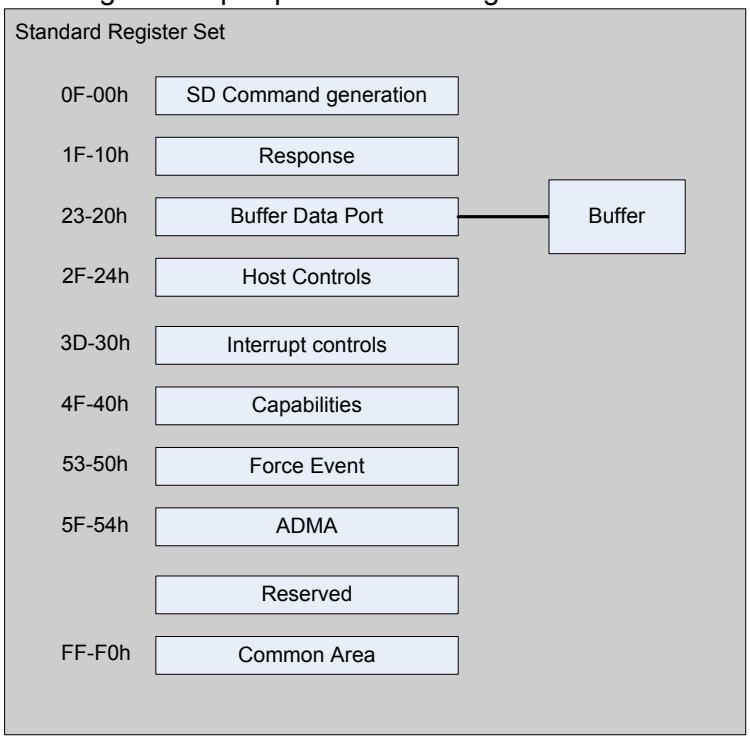

Figure 2.2: SD Host Controller register map

2.1.2 Clock Generator

The logiSDHC supports different kinds of SD Cards (Normal Capacity, High Capacity, different speed grades …). That means that data transfer rate can vary depending on the type of the SD Card connected to the system and the chosen transfer mode.

Therefore, clock signals used in some logiSDHC modules have to be programmable to easily adopt to the desired speed mode.

Clock generator is a module that generates clock signals based on the register settings decoded by host interface.

LogiSDHC also supports power down mode which defines that clocking of some logiSDHC modules should be stopped in order to save on current consumption. At the same time some modules should remain active so that logiSDHC can resume in normal mode when needed. Control of this mechanism is also implemented in clock generator module.

One more feature of the clock generator module is the control of reset mechanism in the logiSDHC IP.

2.1.3 Buffer Controller

The Buffer controller module manages the data flow to and from the data buffer when non DMA transfer is performed. These transfers are defined in a way that the Host Driver accesses (internal) data buffer through 32 – bit **Buffer Data Port** register.

Buffer Controller implements logic that ensures that the access to the buffer is enabled and performed in the desired way. The Standard defines that depending on system bus width (8 – bit, 16 – bit, 32 – bit or 64 - bit) internal data buffer should be accessed in suitable manner (the same data width as it is on the system bus).

In this version of logiSDHC that option is not fully implemented. That means that the data buffer is (from the host system side) always accessed in 32 – bit wide data accesses.

Buffer controller also controls the data flow from the SD Card side by generating address and other control signals for the data buffer side which is accessed from SD Card interface side.

2.1.4 DMA Controller

The DMA controller module manages the data flow to and from the data buffer when DMA transfer is performed.

This module generates all the signals needed to enable data transfer between external memory (via Xylon Memory Bus Interface or AXI4 interface) and the data buffer. It also controls the data flow from the SD Card side by generating address and other control signals for the data buffer side which is accessed from the SD Card interface side.

It is reasonable to assume that not in all logiSDHC applications DMA controller will be needed. This is why it is possible to choose to implement DMA controller or not by setting the VHDL generic parameter.

If the DMA controller is implemented, user can also decide on the type of the DMA controller that better fits the application.

One option is to implement the standard DMA controller as mentioned earlier in text and described in more detail in the chapter Standard DMA controller.

The other option is to implement Xylon – specific DMA controller that is described in more detail in the chapter Xylon – specific DMA controller.

2.1.5 SD Card Interface

The SD Card interface module implements a layer through which the logiSDHC connects to the SD Card. It consists of state – machine which implements the SD communication layer and transforms the host commands into the SD interface signals.

It also fetches the response data from the Card and transfers it to the host interface.

Finally, it also transfers the data from the data buffer to the Card and vice versa.

2.2 PINOUT

Table 2.1: Pinout

Signal	Signal Direction	Description
Memory Interface		
XMB Interface	Bus	Xylon Memory Bus. Refer logiMEM specification
AXI4 Interface	Bus	Refer to AMBA AXI version 4 specification from ARM
Register Interface		
PLBV46 Slave Interface	Bus	Refer to Xilinx-IBM Core connect specification
OPB Slave Interface	Bus	Refer to Xilinx-IBM Core connect specification
AXI4-Lite Interface	Bus	Refer to AMBA AXI version 4 specification from ARM
SD Card ports		
sd_clk	Output	SD Card clock
sd_dat	Input/Output	SD Card data
sd_cmd	Input/Output	SD Card command
Auxiliary signals		
sd_base_lock [1)]	Input	SD base clock DCM lock
sd_base_clk [1)]	Input	SD base clock
sd_int	Output	SD interrupt
sd_wp	Input	SD write protect switch
sd_cd_n	Input	SD card not present
sd_ci_n	Output	SD card not inserted
sd_led_n	Output	LED indicating that SD Card is in use

1 Connection of this port is required for correct logiSDHC operation

3 CLOCK SIGNALS

There are three clock domains that can be identified in logiSDHC.
Host system interface related circuits are synchronous to register clock signal. Depending on the register interface that is implemented, the register clock signal is either: opb_clk, plb_clk or s_axi_aclk.
DMA transfer related circuits are synchronous to memory clock signal (mclk for XMB and m_axi_aclk for AXI4 memory interafce). All the SD Card interface related circuits are synchronous to sd_clk clock signal.Signal sd_clk is derived from the sd_base_clk signal in the clock generator module. The frequency of the sd_clk signal is therefore equal to the frequency of the sd_base_clk signal divided by the prescaler value as explained in the chapter Clock Control Register. In order to support the maximum transfer rate defined by the specification (25 MB/s), sd_base_clk frequency should equal 100 MHz.

Clock domain crossing is implemented through true dual port FIFO buffer and handshaking (request ⇔ acknowledge mechanism) and therefore the mentioned clock signals are entirely independent (not correlated).

The only exception is the requirement that the frequency of the sd_clk signal should always be smaller than the frequency of the register and memory clock signal.
This means that, in order to support the highest frequency of the sd_clk (50 MHz) defined by the Standard, the frequency of the register and memory clock should be greater than 50 MHz. This requirement is however most likely met in most applications by default.
Register and memory clock signals are entirely independent.

4 MEMORY INTERFACE

The logiSDHC memory interface is implemented as either XMB (Xylon Memory Bus) master interface or AMBA AXI4 master interface. XMB interface is primarily designed for use with logiMEM – Flexible SDR/DDR memory controller, a member of Xylon's IP library named the logicBRICKS™.
The logiMEM and other Xilinx memory controllers support the UMA (Unified Memory Architecture). The UMA architecture has a number of sub-systems that share a common memory (usually SDRAM memory chips).
logiSDHC XMB master interface width is defined by C_MEM_DATA_BUS_WIDTH parameter while AXI4 has its own parameters. In this version of the logiSDHC only the 32 – bit wide interface is supported. For better memory bandwidth utilization burst accesses are used. XMB interface memory burst can be changed by setting the VHDL generic parameter C_MEM_BURST. In that way 16, 32 and 64 cycle burst accesses are supported.
For more information on XMB bus architecture refer to Xylon logiMEM documentation and for more information on AMBA AXI4 bus architecture refer to bus specification from ARM.

5 HOST SYSTEM INTERFACE

5.1 REGISTER INTERFACE

The logiSDHC registers are accessed thru Slave OPB, Slave PLBv4.6 or Slave AXI4-Lite interface depending on VHDL generic parameter C_REGS_INTERFACE. Please consult Xilinx OPB or PLBv4.6 specification or specification on AMBA AXI4 from ARM regarding the functionality of these interfaces.
All registers inside logiSDHC can be read but not all can be written (check chapter REGISTERS), however some registers have different write and read values. Register map is completely defined by the Standard so some bits have reserved values. Variable register widths are also defined so that the 8, 16 or 32 bit wide registers are used. Non used bits inside each register are reserved or read '0'. Overview of register memory map is given in chapter REGISTERS and detailed description is given in the Standard.
OPB, SPLB or S_AXI clocks are independent of other clocks connected to logiSDHC.

5.2 INTERRUPT INTERFACE

The SD Host Controller interrupt interface and functionality is defined by the Standard.
When the interrupt source is active than both interrupt status and/or signal are activated, provided that the interrupt functionality is enabled in the corresponding interrupt enable registers as explained In the chapter

REGISTER DESCRIPTION.

The interrupt signal (sd_int) is level sensitive (active high).
Detailed description of all interrupt sources can be found in the explanation of the corresponding interrupt registers as described in chapter REGISTERS.

6 REGISTERS

The logiSDHC registers are accessed through the OPB slave, Slave PLBv4.6 or Slave AXI4-Lite interface by the Microblaze, PowerPC or any other OPB/PLB/AXI master (logiSPI, Ext CPU OPB/PLB/AXI master…).

6.1 REGISTER MAP

All registers are placed at an offset set by VHDL generic parameter C_REGS_BASEADDR.
The following table describes the names and addresses of logiSDHC registers on the OPB/PLB/AXI bus.
The following abbreviations are being used to describe the type of the register:

- R/W => read or write
- ROC => read only status: initialized to zero at reset, write to these bits is ignored
- RO => read only: cannot be altered by software or any reset operation, write to these bits is ignored
- Rsvd => Reserved: initialized to zero and writes to them are ignored

Table 6.1 logiSDHC register map

Address offset	Type	Size (bits)	Name	Description
0x0000	R/W	32	SDMA_ADDR	SDMA System Address Register
0x0004	R/W	16	BLOCK_SIZE	Block Size Register
0x0006	R/W	16	BLOCK_COUNT	Block Count Register
0x0008	R/W	32	ARGUMENT_REG	Argument Register
0x000C	R/W	16	TRANSFER_MODE	Transfer Mode Register
0x000E	R/W	16	COMMAND_REG	Command Register
0x0010	R/W	32	RESPONSE_REG0	Response Register bits 0 – 31
0x0014	R/W	32	RESPONSE_REG1	Response Register bits 32 – 63
0x0018	R/W	32	RESPONSE_REG2	Response Register bits 64 – 95
0x001C	R/W	32	RESPONSE_REG3	Response Register bits 94 – 127
0x0024	R/W	32	PRES_STAT_REG	Present State Register
0x0028	R/W	8	HOST_CTRL_REG	Host Control Register
0x0029	R/W	8	POWER_REG	Power Control Register
0x002A	R/W	8	BLOCK_GAP_REG	Block Gap Register
0x002B	R/W	8	WAKEUP_REG	Wakeup Register
0x002C	R/W	16	CLOCK_CTRL_REG	Clock Control Register
0x002E	R/W	8	TIMEOUT_REG	Timeout Register
0x002F	R/W	8	SOFT_RST_REG	Software Reset Register
0x0030	R/W	16	INT_REG	Normal Interrupt Status Register
0x0032	R/W	16	ERROR_INT_REG	Error Interrupt Status Register [1]
0x0034	R/W	16	INT_EN_REG	Normal Interrupt Status Enable Register
0x0036	R/W	16	ERROR_INT_EN_REG	Error Interrupt Status Enable Register [1)]
0x0038	R/W	16	INT_SIG_EN_REG	Normal Interrupt Signal Enable Register
0x003A	R/W	16	ERROR_INT_SIG_EN_REG	Error Interrupt Signal Enable Register

Address offset	Type	Size (bits)	Name	Description
0x003C	R/W	16	AUTO_CMD12_REG	Auto CMD12 Error Status Register
0x0040	R/W	32	CAPABILITIES_REG0	Capabilities Register bits 0 – 31
0x0044	R/W	32	CAPABILITIES_REG1	Capabilities Register bits 32 – 63
0x0048	R	32	MAX_CURRENT_REG	Maximum Current Capabilities Register
…	…	…	…	Reserved
…	…	…	…	Reserved
…	…	…	…	Reserved
…	…	…	…	Reserved
0x00F0	W	16	PIX_START_BYTE	Non Standard [2]
0x00F2	W	16	ALPHA_REG	Non Standard [2]
0x00F4	W	16	HRES_REG	Non Standard [2]
0x00F6	W	16	VRES_REG	Non Standard [2]
0x00FC	R	16	SLOT_INTERRUPT_STAT	Slot Interrupt Status Register
0x00FE	R	16	HOST_VERSION_REG	Host Controller Version Register

NOTE: Detailed register description can be found in SD Host Controller Simplified Specification Version 2.00

[1] Vendor specific error status bit added, check in Vendor specific status bits.
[2] non Standard refers to Xylon – specific DMA controller. More detailed description of these registers is available in the following section.

6.2 REGISTER DESCRIPTION

6.2.1 SDMA System Address Register

Table 6.2: SDMA System Address Register Description

Bits	Name	Type	Reset value	Description
31-00	SDMA_REG	R/W	0	**System memory address** for SDMA transfer

This register contains the physical system memory address used for SDMA transfer. The actual address set on the XMB bus at the beginning of the DMA transfer equals the value written in this register added to the value set in the generic parameter C_MEM_BASEADDR.
The SDMA transfer waits at every boundary specified by the Host SDMA Buffer Boundary in the BLOCK_SIZE_REG register. The Host Controller generates DMA Interrupt to request the Host Driver to update this register. The Host Driver sets the next system address of the next data position to this register. After the most upper byte of this register (003h) is written, the SDMA transfer continues.

6.2.2 Block Size Register

Values written in this register define the transfer parameters:

- Host buffer boundary for SDMA transfers
- Transfer block size

Table 6.3: Block Size Register Description

Bits	Name	Type	Reset value	Description
15		Rsvd		Reserved
14-12	SDMA Buffer Boundary	R/W	000	**Host SDMA Buffer Boundary**. The large contigous memory space may not be available in the memory system. To perform long SDMA transfer, SDMA System Address register shall be updated at every system memory boundary during SDMA transfer. Values: • 000b - 4K bytes • 001b - 8K bytes • 010b - 16K bytes • 011b - 324K bytes • 100b - 64K bytes • 101b - 128K bytes • 110b - 256K bytes • 111b - 512K bytes
11-00	Transfer Block Size	R/W	0000h	**Transfer Block Size** Specifies the block size of data transfers. Values: • 0800h - 2048 Bytes • … - … • 0200h - 512 Bytes • 01FFh - 511 Bytes • … - … • 0004h - 4 Bytes • 0003h - 3 Bytes • 0002h - 2 Bytes • 0001h - 1 Byte • 0000h – no data transfer

6.2.3 Block Count Register

Value written in this register defines the number of the block to be transferred in the transfer being initialized.

Table 6.4: Block Count Register Description

Bits	Name	Type	Reset value	Description
15-00	Block_cnt	R/W	0000h	**Blocks Count** for current transfer. This register is enabled when *Block Count Enable* in the *TRANSFER_MODE* register is set to 1 and is valid only for multiple block transfers. Values: • FFFFh– 65535 blocks • … – … • 0002h – 2 blocks • 0001h – 1 block • 0000h – Stop Count

6.2.4 Argument Register

Value written in this register defines the argument of the command being issued to the SD Card.

Table 6.5: Argument Register Description

Bits	Name	Type	Reset value	Description
31-00	Arg_reg	R/W	00000000h	**Command Argument**. The SD Command Argument is specified as bit39-8 of Command-Format in the Physical Layer Specification.

6.2.5 Transfer Mode Register

Values written in this register define the transfer parameters:

- Single or multiple block transfer
- Read or write transfer
- DMA or non DMA transfer

Table 6.6: Transfer Mode Register Description

Bits	Name	Type	Reset value	Description
15-06		Rsvd		Reserved
05	Sd_trans_mult	R/W	0	**Multi/Single Block Select** This bit is set (=1) when issuig multiple-block command for all other commands it should be set to 0.
04	Sd_trans_dir	R/W	0	**Data Transfer Direction Select** Values: • 1– Read (Card to Host) • 2– Write (Host to Card)
03		Rsvd		Reserved
02	CMD12_en	R/W	0	**Auto CMD12 Enable** Multiple block transfers require CMD12 to stop the

Bits	Name	Type	Reset value	Description
				transaction. When this bit is set to 1 the Host Controller shall issue CMD12 automatically when the last block is transferred. [1]
01	Block_cnt_en	R/W	0	**Block Count Enable** Enable bit for the *BLOCK_COUNT* register, only relevant for multiple block transfers. Values: • 0– Disable • 1– Enable
00	Dma_en	R/W	0	**DMA Enable** This bit enables the DMA functionality. Values: • 0– Disable • 1– Enable

[1] AutoCMD12 functionality is not supported in this version of the logiSDHC as noted in the chapter AutoCMD12.

6.2.6 Command Register

Values written in this register define the parameters of the command being issued to the SD Card.

Table 6.7: Command Register Description

Bits	Name	Type	Reset value	Description
15-14		Rsvd		Reserved
13-08	Cmd_index	R/W	000000b	**Command Index**
07-06	Cmd_type	R/W	00b	**Command type** There are three types of special commands: Suspend, Resume and Abort. For all other commands this bits should be set to 0. Values: • 11b – Abort • 10b – Resume • 01b – Suspend • 00b – Normal
05	Data_present	R/W	0	**Data Present Select** This bit is set to indicate that the transfer using the DAT line is about to start.
04	Cmd_index_c	R/W	0	**Command Index Check Enable** When this bit is set to 1, the Host Controller checks the index field in the response to see if the command index error occurred and reports the status.
03	Cmd_crc_c	R/W	0	**Command CRC Check Enable** When this bit is set to 1, the Host Controller checks the CRC field in the response to see if the command CRC error occurred and reports the status.
02		Rsvd		Reserved
01-00	Resp_type	R/W	00b	**Response Type Select** Values:

Bits	Name	Type	Reset value	Description
				• 00b – No Response • 01b – Response Length 136 • 10b – Response Length 48 • 11b – Response Length 48 check busy after response

6.2.7 Response register

Values read from this register represent the SD Card response data.

Table 6.8: Response Register Description

Bits	Name	Type	Reset value	Description
127-00	Resp_reg	ROC		**Command Response**

6.2.8 Buffer Data Port Register

The data buffer is accessed (read or written) through this register (for non DMA transfers).

Table 6.9: Buffer Data Port Register Description

Bits	Name	Type	Reset value	Description
31-00	Data_buf	R/W	0	**Buffer Data** The Host Controller Buffer is accessed through this 32-bit register when non-DMA data transfer is performed.

6.2.9 Present State Register

Bits in this register define the actual state and status of the logiSDHC functions.

Table 6.10: Present State Register Description

Bits	Name	Type	Reset value	Description
31-25		Rsvd		Reserved
24	CMD_line_l	RO	0	**CMD Line Signal Level** Used to check the **CMD** line level to recover from errors and for debugging.
23-20	DAT_line	RO	0	**DAT[3:0] Line Signal Level** Used to check the DAT line level to recover from errors, detecting busy signal and debugging. Values: • D23 – DAT[3] • D22 – DAT[2] • D21 – DAT[1] • D20 – DAT[0]
19	WP_switch	RO	0	**Write Protect Switch Pin Level** Reflects the SDWP pin. Values: • 1 – write enabled (SDWP = 1) • 0 – write protected (SDWP = 0)
18	Card_detect	RO	0	**Card Detect Pin Level** Reflects the SDCD pin. Values: • 1 – card present (SDCD = 1) • 0 – No card present (SDCD = 0)
17	Card_state	RO	1	**Card State Stable** This bit is used for testing. Values: • 1 – No card or Inserted • 0 – Reset or Debouncing
16	Card_inserted	RO	1	**Card Inserted** Values: • 1 – Card Inserted • 0 – Reset or Debouncing or No Card
15-12		Rsvd		Reserved
11	Buf_read_en	ROC	0	**Buffer Read Enable** Used for non-DMA transfers. Indicates that one data block has been read from the Card and is ready to be transferred from the data buffer. Values: • 1 – Read enable • 0 – Read disable
10	Buf_write_en	ROC	0	**Buffer Write Enable** Used for non-DMA transfers. Indicates that the data buffer is ready to be written. Values: • 1 – Write enable • 0 – Write disable
09	Read_trans_a	ROC	0	**Read Transfer Active**

Bits	Name	Type	Reset value	Description
				Used for non-DMA transfers to detect the completion of a read transfer. Values: • 1 – Transferring data • 0 – No valid data
08	Write_trans_a	ROC	0	**Write Transfer Active** Used for non-DMA transfers to indicate that a write transfer is active. Values: • 1 – Transferring data • 0 – No valid data
07-03		Rsvd		Reserved
02	Dat_line_a	ROC	1	**DAT Line Active** Indicates that one of the DAT lines on the SD Bus is in use. Values: • 1 – DAT Line Active • 0 – DAT Line Inactive
01	Inhibit_DAT	ROC	0	**Command Inhibit(DAT)** This bit is set if either the DAT Line Active or Read Transfer Active is set to 1. Values: • 1 – Cannot issue command which uses the DAT line • 0 – Can issue command which uses the DAT line
00	Inhibit_AMD	ROC	0	**Command Inhibit(CMD)** This bit is set when the CMD line is in use and the Host Controller cann not issue another SD Command using the CMD line. Values: • 1 – Cannot issue command • 0 – Can issue command

6.2.10 Host Control Register

The bits in this register define current logiSDHC properties:

Table 6.11: Host Control Register Description

Bits	Name	Type	Reset value	Description
07	Card_det_sel	R/W	0	**Card Detect Signal Selection** This bit selects the source of the card detect signal. Values: • 1 – The Card Detect Test Level is selected (for test purpose) • 0 – SDCD is selected (for normal use)
06	Card_det_l	R/W	0	**Card Detect Test Level** Enabled while the Card Detect Signal Selection is set to 1 and indicates if the Card is selected. Values: • 1 – Card Inserted • 0 – No card
05		Rsvd		Reserved
04-03	DMA_sel	R/W	1	**DMA Select** Values: • 00 – SDMA selected • 01 – Reserved • 10 – 32-bit address ADMA2 selected • 11 – 64-bit address ADMA2 selected
02	High_speed_e	R/W	0	**High Speed Enable** Values: • 1 – High speed mode • 0 – Normal mode
01	Trans_width	R/W	0	**Data Transfer width** Values: • 1 – 4-bit mode • 0 – 1-bit mode
00	LED_ctrl	R/W	0	**LED Control** Indicates that the Card is in use and it shouldn't be removed. Values: • 1 – LED on • 0 – LED off

6.2.11 Power Control Register

Bits in this register define SD Bus power control properties:

Table 6.12: Power Control Register Description

Bits	Name	Type	Reset value	Description
07-04		Rsvd		Reserved
03-01	Bus_voltage	R/W	0	SD Bus Voltage Select Values: • 111b – 3.3V (Typ.) • 110b – 3.0V (Typ.) • 101b – 1.8V (Typ.) • 100b – 000b – Reserved
00	Bus_power	R/W	0	SD Bus Power Values: • 1 – power on • 0 – power off

6.2.12 Block Gap Control Register

This register is implemented but the Block Gap (stop - resume) mechanism is not supported in this version of the logiSDHC, as stated in the chapter SPECIFICATION EXCEPTIONS.

6.2.13 Wakeup Control Register

This register is implemented but the wakeup functionality is not supported in this version of the logiSDHC, as stated in the chapter SPECIFICATION EXCEPTIONS.

6.2.14 Clock Control Register

Bits in this register define the sd_clk generation and control properties:

Table 6.13: Clock Control Register Description

Bits	Name	Type	Reset value	Description
15-08	Sd_freq_sel	R/W	0	**SDCLK Frequency Select** Defines the frequency of the sd_clk. Values: • 80h – sd_base_clk divided by 512 • 40h – sd_base_clk divided by 256 • 20h – sd_base_clk divided by 128 • 10h – sd_base_clk divided by 64 • 08h – sd_base_clk divided by 32 • 04h – sd_base_clk divided by 16 • 02h – sd_base_clk divided by 8 • 01h – sd_base_clk divided by 4 • 00h – sd_base_clk divided by 2
07-03		Rsvd		Reserved
02	Sd_clk_en	R/W	1	**SD Clock Enable** Writing 0 to this bit stops the SDCLK. Values: • 1 – Enable • 0 – Disable
01	Int_clk_stable	ROC	1	**Internal Clock Stable** Set to 1 when the sd_clk is stable. Host Driver should set the int_clk_en bit in this register and then wait until this bit is set to 1 before it enables the sd_clk by writing 1 to the sd_clk_en bit in this register. Values: • 1 – Ready • 0 – Not Ready
00	Int_clk_en	R/W	0	**Internal Clock Enable** Set to 0 when the Host Driver doesn't use the Host Controller. Values: • 1 – Oscillate • 0 – Stop

6.2.15 Timeout Control Register

The value written to this register determines the interval by which DAT line timeouts are detected. DAT line timeout detection functionality is not implemented in this version of the logiSDHC as stated in SPECIFICATION EXCEPTIONS.

Writing to or reading from this register therefore doesn't have the real meaning.

6.2.16 Software Reset Register

A reset pulse is generated after writing 1 to each bit of this register. After completing the reset the Host Controller clears each bit.

Table 6.14: Software Reset Register Description

Bits	Name	Type	Reset value	Description
07-03		Rsvd		Reserved
02	Rst_DAT	R/W	0	**Software Reset For DAT Line**
01	Rst_CMD	R/W	0	**Software Reset For CMD Line**
00	Rst_all	R/W	0	**Software Reset For All**

6.2.17 Normal Interrupt Status Register

Bits in this register represent the status of the normal interrupt sources:

Table 6.15: Normal Interrupt Status Register Description

Bits	Name	Type	Reset value	Description
15	Error_int	ROC	0	**Error Interrupt** Set if any bit in the *ERROR_INT_STATUS_REG* is set. Values: • 1 – Error • 0 – No Error
14-09		Rsvd		Reserved
08	Card_int	ROC	0	**Card Interrupt** Writing 1 to this bit doesn't cleare the bit. It is cleared only when the source of the Card Interrupt is removed. Values: • 1 – Generate Card Interrupt • 0 – No Card Interrupt
07	Card_removal	R/W	0	**Card Removal** This bit is set if the *Card_inserted* in the *PRESENT_STAT_REG* changes from 1 to 0. Values: • 1 – Card Removed • 0 – Card State stable or Debouncing
06	Card_insert	R/W	0	**Card Insertion** This bit is set if the *Card_inserted* in the *PRESENT_STAT_REG* changes from 0 to 1. Values: • 1 – Card Inserted • 0 – Card State stable or Debouncing
05	Buf_read_rdy	R/W	0	**Buffer Read Ready** This bit is set if the *buf_read_en* in the *PRESENT_STAT_REG* changes from 0 to 1. Values: • 1 – Ready to read buffer • 0 – Not ready to read buffer
04	Buf_write_rdy	R/W	0	**Buffer Write Ready** This bit is set if the *buf_write_en* in the

Bits	Name	Type	Reset value	Description
				PRESENT_STAT_REG changes from 0 to 1. Values: • 1 – ready to write buffer • 0 – not ready to write buffer
03	DMA_int	R/W	0	**DMA Interrupt** Set when the Host Controller detects the Host SDMA Buffer boundary during the transfer. Values: • 1 – DMA Interrupt is generated • 0 – no DMA Interrupt
02	Block_gap_e	R/W	0	**Block Gap Event** The block gap functionality is not implemented. See SPECIFICATION EXCEPTIONS
01	Trans_cmpl	R/W	0	**Transfer Complete** Set when the read or write transfer is done. Values: • 1 – Command execution is completed • 0 – not complete
00	Cmd_cmpl	R/W	0	**Command Complete** Set when the get the end bit of the command response. Values: • 1 – Command completed • 0 – no command complete

6.2.18 Error Interrupt Status Register

Bits in this register represent the status of the error interrupt sources:

Table 6.16: Error Interrupt Status Register Description

Bits	Name	Type	Reset value	Description
15-13	Vend_spec	R/W	0	Vendor Specific Error Status Not used.
12	Buf_overflow	R/W	0	**Buffer Overflow** Vendor Specific status bit. For more detail check Vendor specific status bits. Values: • 1 – Host slower than the SD interface • 0 – transfer runs at the full speed
11-10		Rsvd		Reserved
09	ADMA_err	R/W	0	**ADMA Error** ADMA functionality not implemented in this version of the logiSDHC. For more detail check SPECIFICATION EXCEPTIONS.
08	Auto_CMD12	R/W	0	**Auto CMD12 Error** Auto CMD12 functionality not implemented in this version of the logiSDHC. For more detail check SPECIFICATION EXCEPTIONS.
07	Current_err	R/W	0	**Current Limit Error** This bit is not implemented in this version of the

Bits	Name	Type	Reset value	Description
				logiSDHC.
06	Data_end_err	R/W	0	**Data End Bit Error** Values: • 1 – Error • 0 – No Error
05	Data_crc_err	R/W	0	**Data CRC Error** Values: • 1 – Error • 0 – No Error
04	Data_tout_err	R/W	0	**Data Timeout Error** Values: • 1 – Error • 0 – No Error
03	Cmd_ind_err	R/W	0	**Command Index Error** Values: • 1 – Error • 0 – No Error
02	Cmd_ebit_err	R/W	0	**Command End Bit Error** Values: • 1 – Error • 0 – No Error
01	Cmd_CRC	R/W	0	**Command CRC Error** Values: • 1 – Error • 0 – No Error
00	Cmd_tout_err	R/W	0	**Command Timeout Error** Values: • 1 – Error • 0 – No Error

6.2.19 Normal Interrupt Status Enable Register

Bits in this register represent the status enable bits of the normal interrupt sources:

Table 6.17: Normal Interrupt Status Enable Register Description

Bits	Name	Type	Reset value	Description
15		RO	0	Fixed to 0
14-09		Rsvd		Reserved
08	Card_int_en	R/W	0	**Card Interrupt Status Enable** Values: • 1 – Enabled • 0 – Masked
07	Card_rem_en	R/W	0	**Card Removal Status Enable** Values: • 1 – Enabled • 0 – Masked
06	Card_ins_en	R/W	0	**Card Insertion Status Enable** Values: • 1 – Enabled • 0 – Masked
05	Buf_read_en	R/W	0	**Buffer Read Ready Status Enable** Values: • 1 – Enabled • 0 – Masked
04	Buf_wr_en	R/W	0	**Buffer Write Ready Status Enable** Values: • 1 – Enabled • 0 – Masked
03	DMA_int_en	R/W	0	**DMA Interrupt Status Enable** Values: • 1 – Enabled • 0 – Masked
02	Block_gap_en	R/W	0	**Block Gap Event Status Enable** The block gap functionality is not implemented. See SPECIFICATION EXCEPTIONS
01	Trans_cpl_en	R/W	0	**Transfer Complete Status Enable** Values: • 1 – Enabled • 0 – Masked
00	Cmd_cpl_en	R/W	0	**Command Complete Status Enable** Values: • 1 – Enabled • 0 – Masked

6.2.20 Error Interrupt Status Enable Register

Bits in this register represent the status enable bits of the error interrupt sources:

Table 6.18: Error Interrupt Status Enable Register Description

Bits	Name	Type	Reset value	Description
15-13	Vendor_spec	R/W	0	Vendor Specific Status Not implemented
12	Buf_overfl_en	R/W	0	**Buffer Overflow Status Enable** Vendor Specific status bit. For more detail check Vendor specific status bits. Values: • 1 – Enabled • 0 – Masked
11-10		Rsvd		Reserved
09	ADMA_err_en	R/W	0	**ADMA Error Status Enable** ADMA functionality not implemented in this version of the logiSDHC. For more detail check SPECIFICATION EXCEPTIONS.
08	CMD12_er_en	R/W	0	**Auto CMD12 Error Status Enable** Auto CMD12 functionality not implemented in this version of the logiSDHC. For more detail check SPECIFICATION EXCEPTIONS.
07	Current_er_en	R/W	0	**Current Limit Error Status Enable** Not implemented in this version of the logiSDHC.
06	Data_end_e	R/W	0	**Data End Bit Error Status Enable** Values: • 1 – Enabled • 0 – Masked
05	Data_CRC_e	R/W	0	**Data CRC Error Status Enable** Values: • 1 – Enabled • 0 – Masked
04	Data_tout_e	R/W	0	**Data Timeuot Error Status Enable** Values: • 1 – Enabled • 0 – Masked
03	Cmd_index_e	R/W	0	**Command Index Error Status Enable** Values: • 1 – Enabled • 0 – Masked
02	Cmd_end_e	R/W	0	**Command End Bit Error Status Enable** Values: • 1 – Enabled • 0 – Masked
01	Cmd_CRC_e	R/W	0	**Command CRC Error Status Enable** Values: • 1 – Enabled • 0 – Masked

00	Cmd_tout_e	R/W	0	**Command Timeout Error Status Enable** Values: • 1 – Enabled • 0 – Masked

6.2.21 Normal Interrupt Signal Enable Register

Bits in this register represent the status enable bits of the normal interrupt signal:

Table 6.19: Normal Interrupt Signal Enable Register Description

Bits	Name	Type	Reset value	Description
15		RO	0	Fixed to 0
14-09		Rsvd		Reserved
08	Card_int_sen	ROC	0	**Card Interrupt Signal Enable** Values: • 1 – Enabled • 0 – Masked
07	Card_re_sen	R/W	0	**Card Removal Signal Enable** Values: • 1 – Enabled • 0 – Masked
06	Card_ins_sen	R/W	0	**Card Insertion Signal Enable** Values: • 1 – Enabled • 0 – Masked
05	Buf_r_rdy_sen	R/W	0	**Buffer Read Ready Signal Enable** Values: • 1 – Enabled • 0 – Masked
04	Buf_w_rdy_se	R/W	0	**Buffer Write Ready Signal Enable** Values: • 1 – Enabled • 0 – Masked
03	DMA_int_sen	R/W	0	**DMA Interrupt Signal Enable** Values: • 1 – Enabled • 0 – Masked
02	Block_gap_e	R/W	0	**Block Gap Event Signal Enable** The block gap functionality is not implemented. See SPECIFICATION EXCEPTIONS
01	Trans_c_sen	R/W	0	**Transfer Complete Signal Enable** Values: • 1 – Enabled • 0 – Masked
00	Cmd_c_sen	R/W	0	**Command Complete Signal Enable** Values: • 1 – Enabled • 0 – Masked

6.2.22 Error Interrupt Signal Enable Register

Bits in this register represent the status enable bits of the error interrupt signal:

Table 6.20: Error Interrupt Signal Enable Register Description

Bits	Name	Type	Reset value	Description
15-13	Vendor_spec	R/W	0	Vendor Specific Status - Not implemented
12	Buf_ovfl_sen	R/W	0	**Buffer Overflow Signal Enable** Vendor Specific status bit. For more detail check Vendor specific status bits. Values: • 1 – Enabled • 0 – Masked
11-10		Rsvd		Reserved
09	ADMA_e_sen	R/W	0	**ADMA Error Signal Enable** ADMA functionality not implemented in this version of the logiSDHC. For more detail check SPECIFICATION EXCEPTIONS.
08	CMD12_e_sen	R/W	0	**Auto CMD12 Error Signal Enable** Auto CMD12 functionality not implemented in this version of the logiSDHC. For more detail check SPECIFICATION EXCEPTIONS.
07	Curr_e_sen	R/W	0	**Current Limit Error Signal Enable** Not implemented in this version of the logiSDHC.
06	Data_end_bit_sen	R/W	0	**Data End Bit Error Signal Enable** Values: • 1 – Enabled • 0 – Masked
05	Data_CRC_e_sen	R/W	0	**Data CRC Error Signal Enable** Values: • 1 – Enabled • 0 – Masked
04	Data_tout_e_sen	R/W	0	**Data Timeuot Error Signal Enable** Values: • 1 – Enabled • 0 – Masked
03	Cmd_index_e_sen	R/W	0	**Command Index Error Signal Enable** Values: • 1 – Enabled • 0 – Masked
02	Cmd_end_e_sen	R/W	0	**Command End Bit Error Signal Enable** Values: • 1 – Enabled • 0 – Masked
01	Cmd_CRC_e_sen	R/W	0	**Command CRC Error Signal Enable** Values: • 1 – Enabled • 0 – Masked
00	Cmd_tout_e_sen	R/W	0	**Command Timeout Error Signal Enable** Values: • 1 – Enabled

Bits	Name	Type	Reset value	Description
				• 0 – Masked

6.2.23 Auto CMD12 Error Status Register

Auto CMD12 functionality is not implemented in this version of the logiSDHC as stated in SPECIFICATION EXCEPTIONS.

6.2.24 PIX_START_BYTE

The value written into the *PIX_START_BYTE* register defines the number of bytes that will be skipped (will not be written to the Data Buffer) from the beginning of the DMA transfer. This feature is a part of Xylon – specific DMA controller mechanism described in the chapter Xylon – specific DMA controller.

Table 6.21: PIX_START_BYTE Register Description

Bits	Name	Type	Reset value	Description
15-00	Pix_start_byte	W	0	**Start Pixel Byte**

6.2.25 ALPHA_REG

The value written into *ALPHA_REG* register defines the alpha value (parameter used for blending) that is added to each pixel transferred to the Data Buffer during the DMA transfer. This feature is a part of Xylon – specific DMA controller mechanism described in the chapter Xylon – specific DMA controller.

Table 6.22: ALPHA_REG Register Description

Bits	Name	Type	Reset value	Description
15-00	Alpha_reg	W	0	**Alpha Register**

6.2.26 HRES_REG

The value written into *HRES_REG* register defines the horizontal resolution of the image that is transferred through the DMA data transfer. This feature is a part of Xylon – specific DMA controller mechanism described in the chapter Xylon – specific DMA controller.

Table 6.23: HRES_REG Register Description

Bits	Name	Type	Reset value	Description
15-00	Hres_reg	W	0	Horisontal Resolution Register

6.2.27 VRES_REG

The value written into *VRES_REG* register defines the vertical resolution of the image that is transferred through the DMA data transfer. This feature is a part of Xylon – specific DMA controller mechanism described in the chapter Xylon – specific DMA controller.

Table 6.24: VRES_REG Register Description

Bits	Name	Type	Reset value	Description
15-00	Vres_reg	W	0	**Vertical Resolution Register**

7 VHDL GENERIC PARAMETERS

The logiSDHC configuration is defined by generics that are configurable thru Xilinx Platform Studio (XPS) user interface. User can set these parameters prior to the IP core's code synthesis to configure the logiSDHC behaviour.

Table 7.1: logiSDHC Parameters

Generic Name	Allowable values	Default value	Description
Version, General Generics			
C_IP_LICENSE_TYPE[1)]	0,1,2	0	Constant: 0 – source, 1 – evaluation, 2 – release
C_IP_MAJOR_REVISION[1)]	0 – 31	1	Constant: Values from 0 to 31
C_IP_MINOR_REVISION[1)]	0 – 31	01	Constant: Values from 0 to 31
C_IP_PATCH_LEVEL[1)]	0 – 25	0	Constant: Values from 0(a) to 25(z)
C_IP_LICENSE_CHECK[1)]	0, 1	0	Constant: 0 – don't check for license, 1 – check for license
Registers, General Generics			
C_REGS_INTERFACE	0,1, 2	2	Registers interface selection (0 – PLB, 1 – OPB, 2 – AXI4-Lite)
Registers, Addresses Generics			
C_REGS_BASEADDR	Valid word aligned address	0xffffffff	Registers base address
C_REGS_HIGHADDR	Valid word aligned address	0x00000000	Registers high address
Registers, OPB Slave Generics			
C_OPB_AWIDTH[2,3)]	32	32	OPB address bus width
C_OPB_DWIDTH[2,3)]	32	32	OPB data bus width
Registers, SPLB Slave v4.6 Generics			
C_SPLB_AWIDTH[4,5)]	32	32	PLB address bus width
C_SPLB_DWIDTH[4,5)]	32	32	PLB data bus width
C_SPLB_MID_WIDTH[4,5)]	1,2,3,4	1	PLB Master ID Bus Width. The value is log2(C_SPLB_NUM_MASTERS)
C_SPLB_NUM_MASTERS[4,5)]			Number of masters that can be connected
Registers, AXI4-Lite Generics			
C_S_AXI_ADDR_WIDTH[6)]	32	32	AXI4-Lite address bus width
C_S_AXI_DATA_WIDTH[6)]	32	32	AXI4-Lite data bus width
C_S_AXI_PROTOCOL[6)]	AXI4LITE	AXI4LITE	Bus protocol supported, non-vhdl constant
Memory interface, General Generics			
C_MEM_INTERFACE[7)]	1, 2	2	Memory interface selection (1 – XMB, 2 – AXI)

Generic Name	Allowable values	Default value	Description
C_MEM_BURST[7)]	4,5,6	4	Memory burst width (4, 5, 6). Memory burst width of 4, 5 or 6 means burst size (num of transfers) of 16, 32 or 64 respectively. Only used with XMB interface.
Memory interface, Addresses Generics			
C_MEM_BASEADDR[7)]	Valid word aligned address	0xffffffff	Memory base address
C_MEM_HIGHADDR[7)]	Valid word aligned address	0x00000000	Memory base address
Memory interface, XMB Generics			
C_MEM_DATA_BUS_WIDTH[7,8)]	32	32	Memory interface data bus width (32)
Memory interface, AXI4 Master Generics			
C_M_AXI_SUPPORTS_THREADS[7,9)]	0	0	Support of threads, non-vhdl constant
C_M_AXI_THREAD_ID_WIDTH[7,9)]	1	1	Width of ID signals for threads
C_M_AXI_SUPPORTS_READ[7,9)]	1	1	Master supports read transactions, non-vhdl constant
C_M_AXI_SUPPORTS_WRITE[7,9)]	0	0	Master supports write transactions, non-vhdl constant
C_M_AXI_SUPPORTS_NARROW_BURST[7,9)]	0	0	Master issues multi-beat transactions in which the size of the data transfers is narrower than the interface data-width, non-vhdl constant
C_M_AXI_DATA_WIDTH[7,9)]	32, 64, 128	64	AXI data bus width
C_M_AXI_ADDR_WIDTH[7,9)]	32	32	AXI address bus width
C_M_AXI_PROTOCOL[7,9)]	AXI4	AXI4	Bus protocol supported, non-vhdl constant
User. DMA Generics			
C_USE_DMA	0,1	1	Implement DMA controller: 0 – no, 1 - yes
C_DMA_TYPE[7)]	0,1	1	DMA controller type: 0 – standard, 1 – Xylon specific
C_ROW_STRIDE[7,10)]	512,1024, 2048	1024	Distance in number of pixels between same colon pixels for adjacent rows
C_BYTE_PER_PIXEL[7,10)]	4	4	Number of bytes per pixel
C_CONVERT_ENDIANNESS[7,10)]	0,1	0	Memory system uses: 0 – no, 1 – yes
User. other Generics			
C_BASE_CLOCK_FREQ[11)]	Integer	100	Indicates the frequency of the sd_base_clk signal. Unit values are in MHz

1. These parameters are constant. Default values depend on the current IP version and purchased license.

2. Valid if C_REGS_INTERFACE = 0
3. These parameters are normally calculated by the XPS based on what devices are connected to the OPB bus
4. Valid if C_REGS_INTERFACE = 1
5. These parameters are normally calculated by the XPS based on what devices are connected to the PLB bus
6. Valid if C_REGS_INTERFACE = 2
7. Valid if C_USE_DMA = 1
8. Valid if C_MEM_INTERFACE = 1
9. Valid if C_MEM_INTERFACE = 2
10. Valid if C_DMA_TYPE = 1
11. For more details about this parameter please check the chapter Clock Control Register

These parameters are available at VHDL Synthesis. The slice count significantly varies depending on these parameters due to the extensive logiSDHC HW configurability. Parameters that are influencing slice count the most are *C_USE_DMA* and *C_DMA_TYPE*.

8 DMA CONTROLLER

8.1 Standard DMA controller

Standard DMA controller is implemented according to the DMA algorithm defined in the SD Host Controller Standard Specification Version 1.00 also called SDMA (Single Operation DMA). This algorithm defines the data transfers where only one SD command transaction is executed per DMA operation.

DMA transfer cycle should be performed as described in the figure below:

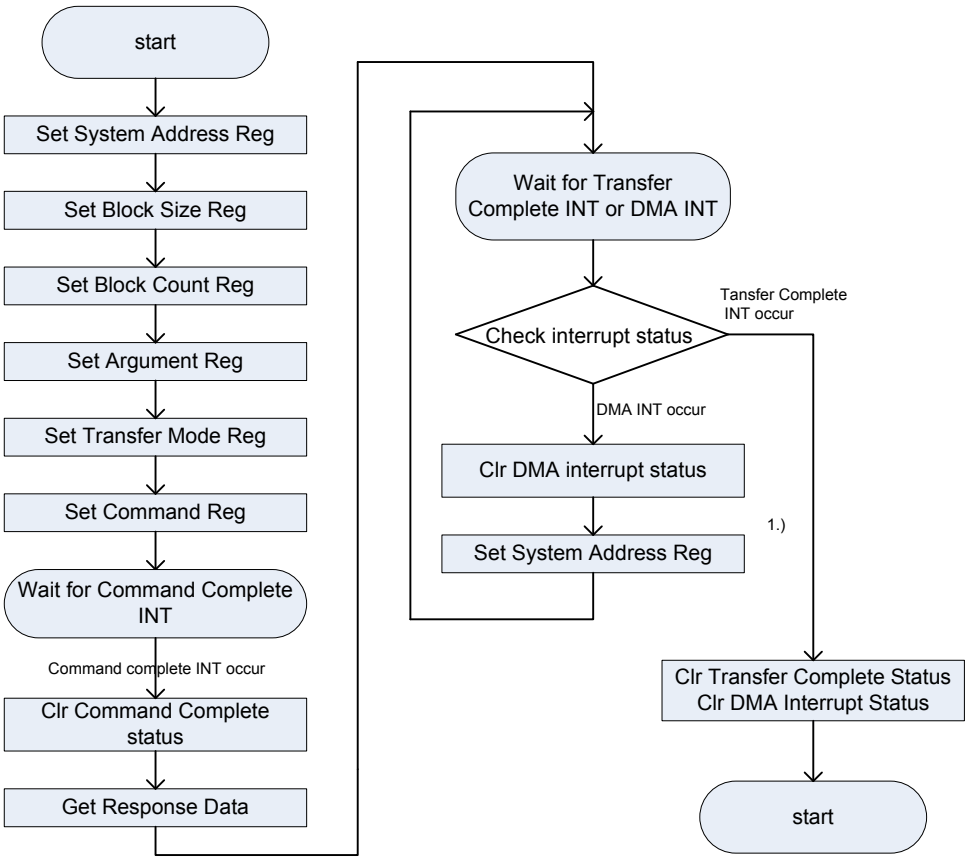

Figure 8.1: SDMA transfer sequence

1) DMA interrupt occurs when the DMA controller detects that the next memory address to be accessed (read or written) exceeds the boundary defined in the *BLOCK SIZE REGISTER*. The transfer is then stopped and it will not continue until the host renews the DMA address (until the most significant byte of the *SYSTEM ADDRESS REG* is written).

8.2 Xylon – specific DMA controller

This DMA controller was designed to support high – speed DMA bitmap transfers and memory block copy transfers from high capacity SD Cards. It is controlled through a set of registers in a way that is closely described in further text.

Since the high – capacity SD cards enable only block data accesses, mechanism in DMA controller is implemented that enables user to specify position inside a block where the transfer data starts. It is depicted in figure below:

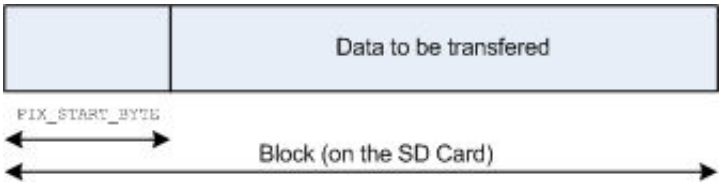

Figure 8.2: PIX_START_BYTE preview

The value written in the *PIX_START_BYTE* register represents the offset (in bytes) from the beginning of the addressed block to the position of the data to be transferred in that DMA transfer.

HRES and *VRES* registers define horizontal and vertical resolution (in pixels), of the image that is being transferred, respectively. C_STRIDE generic VHDL parameter defines raw stride of the destination image. That means that elements of the same column in adjacent rows of the destination image (stored in system memory) will be positioned at C_STRIDE number of bytes distance.

Bitmap images transferred using this DMA controller are very often 24bpps (bit per pixel) RGB images. That format is sometimes not suitable for displaying by Video Controllers so this DMA controller supports reformatting of the transferred image to 32 bpps ARGB image format. This is done by enabling *Insert Dummy Byte* bit (located as bit 8 in the *TRANSFER_MODE* register) as depicted by the figure below:

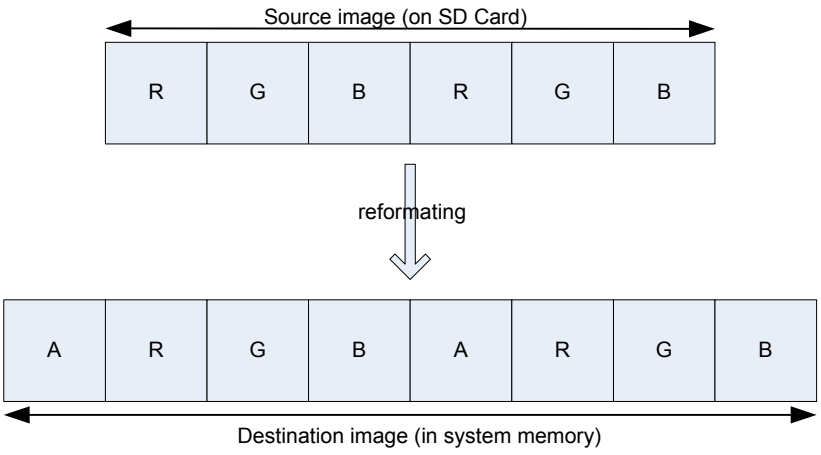

Figure 8.3: Insert Dummy Byte preview

If *Insert Dummy Byte* bit is disabled (= '0') than no reformatting is performed.
Value of the inserted byte is defined by the value written to the *ALPHA_REG* register.
If the C_ENDIAN VHDL parameter was not defined in suitable way during the design development phase that can be easily overcome by setting the *Swap Endian* bit (located as bit 9 in *TRANSFER_MODE* register).

9 SPECIFICATION EXCEPTIONS

9.1 Suspend and Resume mechanism

The Standard specifies that SD transfers can be suspended when needed and resumed after some thread with higher priority (that initiated suspending of SD transfer) has been handled.

This functionality is not implemented with current version of logiSDHC.

9.2 Wakeup functionality

Wakeup functionality is not supported in this version of the logiSDHC.

9.3 DAT line timeout detection

DAT line timeout detection is not supported in this version of the logiSDHC.

9.4 ADMA

The ADMA (Advanced DMA) is DMA algorithm defined in the SD Host Controller Standard Specification 2.00 that adopts scatter gather DMA algorithm, so that higher data transfer speed is available.

This functionality is not implemented with current version of logiSDHC.

9.5 Data Buffer access defined by the system bus

In order to accommodate a variety of system busses, the Standard specifies that 8 – bit, 16 – bit, 32 – bit and 64 – bit accesses to the Data Buffer should be supported.

The current version of logiSDHC supports only 32 – bit accesses to the Data Buffer.

9.6 Multiple slot support

The current version of the logiSDHC supports only one SD Card peripheral, while the Standard proposes that up to 16 SD peripherals can be connected to the Host Controller at the same time.

9.7 AutoCMD12

To end the multiple block data transfer the Host Controller has to issue the CMD12 (stop) command to the SD card. The Standard specifies that the CMD12 is automatically generated by the Host Controller when the CMD12_en bit in the TRANSFER_MODE_REG register is set. However, this functionality is not supported in this version of the logiSDHC and Host Driver therefore needs to take care of issuing the CMD12 after the last data block is transferred.

9.8 Vendor specific status bits

If the Host system (DMA or CPU) is slower than the SD interface (when reading from the card data is written to the buffer faster than it can be transferred further to the host system), buffer overflow occurs. This is detected as buffer overflow error and is signalized to the system by setting the bit 12 in the *ERROR_INT_REG* (when it is enabled as described in *ERROR_INT_REG*).

However, the real meaning of this bit is only to report that the host system is slower than the SD interface and that the transfer is not performed at the full speed enabled by SD interface. Buffer overflow is actually prevented by the mechanism that stops the sd_clk in a way that is suitable to pause the reading from the SD card. The transfer continues when there is enough free space in the buffer for another data block to be read from the SD card.

10 INTEGRATION

This chapter describes how to synthesize the logiSDHC HDL code and integrate the logiSDHC in a host design.

Please skip the chapter "Synthesis" if the logiSDHC gate-level netlist is used. The LogiSDHC is delivered as encrypted source code module, therefore synthesis is always required.

10.1 Synthesis

The logiSDHC source code has configurable register interface, memory data bus width, DMA controller type and many more parameters. Prior to the synthesis, please select the logiSDHC configuration parameters through setting generics using Xilinx EDK tools. See the chapter VHDL GENERIC PARAMETERS for more information.

10.2 Implementation

Recommended logiSDHC FPGA implementation options:

- • Effort: high
- • Pack I/O Registers/Latches into IOB's for: Inputs and Outputs

10.3 Integration

The recommended system integration of the logiSDHC is shown in the image bellow:

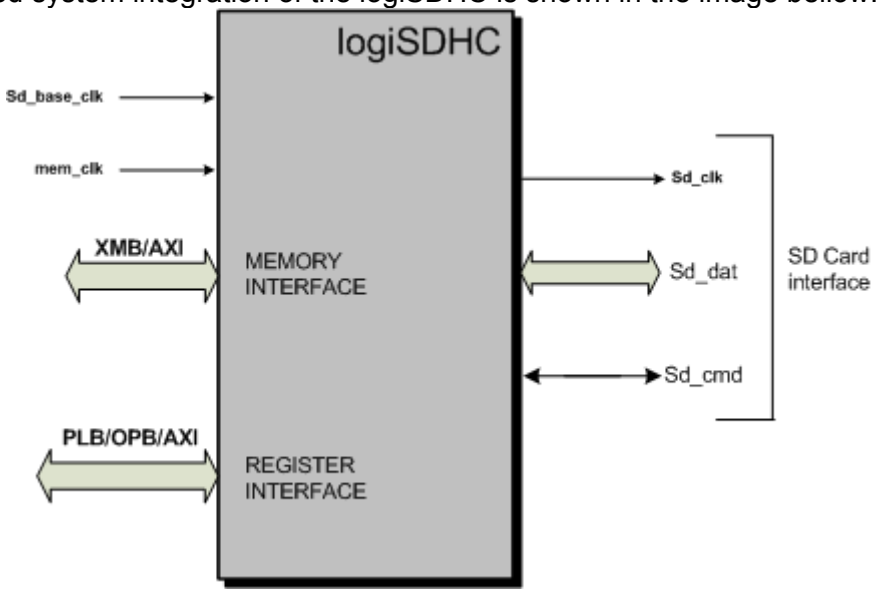

Figure 10.1: logiSDHC system integration

10.4 IO interface description

The logiSDHC - Secure Digital Host Controller is a fully synchronous digital design. Depending on the configuration used, the logiSDHC uses minimum two separated clock signals and maximum three. These clock signals should be connected to the logiSDHC clock inputs *sd_base_clk* for SD interface clock, *mclk* XMB or m_axi_aclk AXI4 clock for memory bus access, *opb_clk* OPB clock, *splb_clk* PLB, *s_axi_aclk* S_AXI clock for registers accesses.

Preferred method of clock generation is by using the Xilinx Virtex-II/Virtex4/Spartan-II/Spartan3 Delay-Locked Loops for global clocks (CLKDLL).

All internal logiSDHC registers are controlled either by the *opb_clk* OPB clock signal, *splb_clk* PLB, or *s_axi_aclk* S_AXI clock signal depending on the used register interface, i.e. VHDL generic parameter C_REGS_INTERFACE.

10.4.1 Input signals

The setup and hold times for FPGA internal FF's is inherently fulfilled by using the Xilinx FPGA implementation tools. The clock skew should be presumed to 0.

Writing and reading to logiSDHC registers is controlled by the OPB/PLB/AXI bus. Please consult Xilinx OPB or PLBv4.6 specification regarding OPB or PLBv4.6 timing requirements or AMBA AXI from ARM.

The memory access is performed through the XMB (Xylon Memory Bus) or AMBA AXI4. Please consult Xylon logiMEM specification regarding timing requirements or AMBA AXI from ARM.

The SD Card interface is implemented in the IO blocks of the FPGA. Depending on the FPGA family used, the appropriated blocks (FDDRRSE, DDR, DDR2 …) are automatically instantiated.

10.5 Timing constraints

The following timing specifications should be appended to the host design user constraint file (UCF).

WARRNING: The logiSDHC related UCF timing constraints can be preceded by the higher priority timing constraints. For example: The PCF timing constraints have the higher priority over UCF constraints. If the logiSDHC related timing constraint is preceded, the implemented design may not work properly due to an erroneous timing constraint.

The lowest clock period, i.e. the highest SD_BASE_CLK frequency period should be supplemented in line:

```
TIMESPEC TS_sd_base_clk = PERIOD "sd_base_clk" <HERE WRITE PERIOD OF sd_base_clk>;
```

Memory clock (XMB or AXI4) and register clock (OPB, PLB or AXI4-Lite) periods have to be supplemented in lines:

```
TIMESPEC TS_mclk = PERIOD "mclk" <HERE WRITE PERIOD OF mclk>;

TIMESPEC TS_m_axi_aclk = PERIOD "m_axi_aclk" <HERE WRITE PERIOD OF m_axi_aclk>;

TIMESPEC TS_opb_clk = PERIOD "opb_clk" <HERE WRITE PERIOD OF opb_clk>;

TIMESPEC TS_splb_clk = PERIOD "splb_clk" <HERE WRITE PERIOD OF splb_clk>;

TIMESPEC TS_s_axi_aclk = PERIOD "s_axi_aclk" <HERE WRITE PERIOD OF s_axi_aclk>;
```

Following lines define constraints important to SD interface:

```
### SD HOST constraints ###

#spartan3:

INST "*logisdhc_0/sdhc_inst/sdhc_card_if_i0/fddrrse_inst.dat_io_gen[?].dat_i_inst"

TNM = "sd_dat_i_int";

INST "*logisdhc_0/sdhc_inst/sdhc_card_if_i0/fddrrse_inst.cmd_i_inst"

TNM = "sd_cmd_i_int";


#virtex4, virtex5, virtex6:

INST "*logisdhc_0/sdhc_inst/sdhc_card_if_i0/ddr_inst.dat_io_gen[?].dat_i_inst"
TNM = "sd_dat_i_int";

INST "*logisdhc_0/sdhc_inst/sdhc_card_if_i0/ddr_inst.cmd_i_inst" TNM = "sd_cmd_i_int";


#spartan3e, spartan6:
```

```
INST
"*logisdhc_0/sdhc_inst/sdhc_card_if_i0/ddr2_inst_S3e_S6.dat_io_gen[?].dat
_i_inst" TNM = "sd_dat_i_int";
INST  "*logisdhc_0/sdhc_inst/sdhc_card_if_i0/ddr2_inst_S3e_S6.cmd_i_inst"
TNM = "sd_cmd_i_int";


#spartan3a, spartan3an:
INST
"*logisdhc_0/sdhc_inst/sdhc_card_if_i0/ddr2_inst.dat_io_gen[?].dat_i_inst
" TNM = "sd_dat_i_int";
INST  "*logisdhc_0/sdhc_inst/sdhc_card_if_i0/ddr2_inst.cmd_i_inst"  TNM  =
"sd_cmd_i_int";


# for all arch:
INST "sd_dat<*>" TNM = PADS "sd_dat";
INST "sd_cmd" TNM = PADS "sd_cmd";
TIMESPEC "TS_sd_dat_in" = FROM "sd_dat" TO "sd_dat_i_int" 4ns;
TIMESPEC "TS_sd_cmd_in" = FROM "sd_cmd" TO "sd_cmd_i_int" 4ns;
INST "sd_clk" IOB = TRUE;
```

The pins should be located and constrained to achieve the best timing properties using FAST and DRIVE constraints, please refer to Xilinx DC and Switching Characteristics user guides for the specific FPGA. If there are no pull-up resistors available on board, you should put PULLUP constraints on dat and cmd pins:

```
# for all arch:
NET "sd_clk" LOC = "<HERE WRITE LOCATION OF sd_clk pin>" | IOSTANDARD =
LVCMOS33 | FAST | DRIVE = 24;
NET " sd_dat<0>" LOC = "<HERE WRITE LOCATION OF sd_dat<0> pin>" | PULLUP
| IOSTANDARD = LVCMOS33 | FAST | DRIVE = 24;
NET " sd_dat<1>" LOC = "<HERE WRITE LOCATION OF sd_dat<0> pin>" | PULLUP
| IOSTANDARD = LVCMOS33 | FAST | DRIVE = 24;
NET " sd_dat<2>" LOC = "<HERE WRITE LOCATION OF sd_dat<0> pin>" | PULLUP
| IOSTANDARD = LVCMOS33 | FAST | DRIVE = 24;
NET " sd_dat<3>" LOC = "<HERE WRITE LOCATION OF sd_dat<0> pin>" | PULLUP
| IOSTANDARD = LVCMOS33 | FAST | DRIVE = 24;
NET " sd_cmd" LOC = "<HERE WRITE LOCATION OF sd_cmd pin>" | PULLUP |
IOSTANDARD = LVCMOS33 | FAST | DRIVE = 24;
```

Revision History

Version	Date	Author	Approved by	Note
1.00.a	15.10.2009	M. Lelas		Initial Xylon release
1.00.b	28.10.2009	M. Lelas		Version update
1.00.c	04.11.2009	M. Lelas		Version update; new document template
1.01.a	29.11.2010	M. Lelas		Version update; new document template
1.05.a	08.06.2011	J. Marjanović		Added AXI support
1.06.a	27.10.2011	J. Marjanović		Version update
1.06.b	15.12.2011	J. Marjanović		Version update
1.06.c	20.01.2012	J. Marjanović		Version update
1.07.a	30.01.2013.	F. Zdunić	R. Končurat	Version update; User's manual document corrections; Ports sd_base_clk and sd_base_lock described as "required"; Changed "Address offset" and "Type" for the following registers: PIX_START_BYTE, ALPHA_REG, HRES_REG, VRES_REG; Removed MAX_CURRENT_REG1; Renamed MAX_CURRENT_REG0 to MAX_CURRENT_REG; Added SLOT_INTERRUPT_STAT to register map; Added C_IP_LICENSE_CHECK generic; Default value for generic parameter C_REGS_INTERFACE set to AXI4-Lite; Default value for generic parameter C_MEM_INTERFACE set to AXI4