International Journal of Electronics Signals and Systems

Volume 3 | Issue 4 Article 11

April 2014

FUNCTIONAL VERIFICATION OF SECURE DIGITAL HOST CONTROLLER

L. JAIKIRAN
ECE Dept., R V College of Engineering, R V Vidyanikethan Post Mysore Road, Bangalore-560059, India, jaikiran.030@gmail.com

PRAKASH BISWAGAR
ECE Dept., R V College of Engineering, R V Vidyanikethan Post Mysore Road, Bangalore-560059, India, prakash.biswagar@rvce.edu.in

THEIVENTHIRAN MURUGAN
Adventura Technologies Pvt. Ltd, koramangala, Bangalore, India, theiven@arasan.com

Recommended Citation
JAIKIRAN, L.; BISWAGAR, PRAKASH; and MURUGAN, THEIVENTHIRAN (2014) "FUNCTIONAL VERIFICATION OF SECURE DIGITAL HOST CONTROLLER," *International Journal of Electronics Signals and Systems*: Vol. 3 : Iss. 4 , Article 11.
DOI: 10.47893/IJESS.2014.1184
Available at: https://www.interscience.in/ijess/vol3/iss4/11

FUNCTIONAL VERIFICATION OF SECURE DIGITAL HOST CONTROLLER

JAIKIRAN L[1], PRAKASH BISWAGAR[2] & THEIVENTHIRAN MURUGAN[3]

[1&2]ECE Dept., R V College of Engineering, R V Vidyanikethan Post Mysore Road, Bangalore-560059, India
[3]Adventura Technologies Pvt. Ltd, koramangala, Bangalore, India
E-mail: jaikiran.030@gmail.com[1], prakash.biswagar@rvce.edu.in[2], theiven@arasan.com[3]

Abstract– Portable storage devices are becoming popular and growing rapidly. These devices can store and acquire information wherever whenever you need. The important applications of portable storage devices are to make backup copies of important data, to share information between different computers or persons, to store digital pictures, music, games, power point presentations etc., and to secure information. These devices are very cost effective, easy to use, and extremely practical.

Keywords *- SD Host Controller, DMA, ADMA, Verilog HDL.*

I. INTRODUCTION

With the increasing consumer digital content, demand for high capacity digital storage is increasing rapidly. Today, portable storage media's are widely used in all mobile phones, digital cameras, camcorders, and in many multimedia devices. Different memory formats like Flash, Secure Digital (SD), Compact Flash, Universal Serial Bus (USB), and Multimedia Card (MMC) are available in the market to store the digital contents. Of all these formats, SD provides many advantages over other formats. Secure Digital (SD) is a Non-volatile memory card format developed by the SD CARD Association (SDA) for use in portable devices. SD cards provide high storage capacity, higher transfer speed, and interoperability with Personal Computer (PC) - related devices and multimedia products.

It is the leading standard for mobile phones, digital cameras, audio players, personal computers, printers, car navigation systems, electronic books, and many other consumer electronic devices. The previous versions of SD Host Controller were having a bus speed of upto 25Mbps. The Host Controller version 3.0 operates in UHS-I and provides bus speed up to 104Mbps.

SD standards are available in three capacity formats: SD, SDHC and SDXC. The Standard-Capacity (SDSC) card family, commonly termed SD, has an official maximum capacity of 2 GB, though some are available up to 4 GB. The High-Capacity (SDHC) card family has a capacity of 4 GB to 32 GB, eXtended-Capacity (SDXC) card family have a capacity starting above 32 GB with a maximum of 2 TB.

The standards have the broad interoperability and compatibility needed to ensure support between multiple devices and for future applications.

II. ARCHITECTURE OF SD HOST CONTROLLER

The SD3.0 / SDIO3.0 / eMMC4.5 Host Controller (3MCR Host Controller) is a Host Controller with an ARM processor interface. This product conforms to SD Host Controller Standard Specification Version 3.00. The block diagram of Host Controller is shown below in figure 1.

The Host Controller handles SDIO/SD Protocol at transmission level, packing data, adding cyclic redundancy check (CRC), Start/End bit, and checking for transaction format correctness. The Host Controller provides Programmed IO method and DMA data transfer method. In programmed IO method, the ARM processor transfers data using the Buffer Data Port Register.

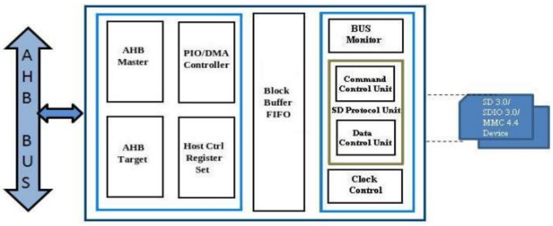


Fig.1: Block Diagram of SD Host Controller

Host controller support for DMA can be determined by checking the DMA support in the Capabilities register. DMA allows a peripheral to read or write memory without the intervention from the CPU. The Host Controller's AHB Host Controller system address register points to the first data address, and data is then accessed sequentially from that address.

A. *Host Controller:*

The Host Controller comprises of Host AHB interface, Host controller registers, Bus Monitor, Clock Generator, and CRC Generator and checker.

The host AHB interface acts as the bridge between AHB and Host Controller. The SD/SDIO controller registers are programmed by the ARM Processor through AHB slave interface. Interrupts are generated to the ARM Processor based on the values set in the Interrupt status register and Interrupt enable registers. Bus monitor will check for any violations occurring in the SD bus and time-out conditions. The Clock generation block will generate the SD clock depending on the value programmed by the ARM Processor in the Clock Control Register. The CRC7 and CRC16 generators calculate the CRC for command and Data respectively to send the CRC to the SD/SDIO and eMMC card. The CRC7 and CRC16 checker check for any CRC error in the Response and Data sent by the SD/SDIO card.

Register Map:
The standard register map is classified in 12 parts listed below. The Host Controller shall support byte, word and double word accesses to these registers. Reserved bits in all registers shall be fixed to zero. The Host Controller shall ignore writes to reserved bits; however, the Host Driver should write them as zero to ensure compatibility with possible future revisions to this Specification.

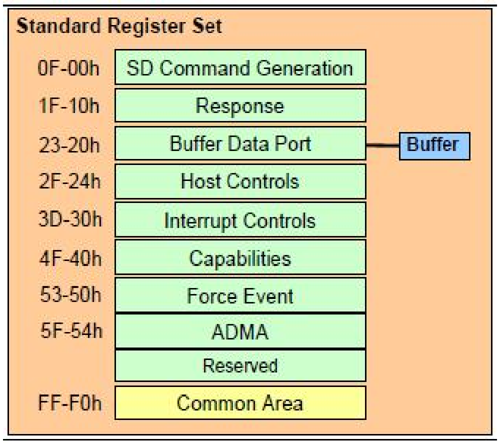


Fig. 2 : SD Standard Register Map

B. *Data FIFO*
The SD/SDIO Host Controller uses one 1k dual port fifo for performing both read and write transactions. During a write transaction (data transferred from ARM Processor to SD3.0 / SDIO3.0 / eMMC4.41 card), the data will be filled in to the first and second half of the FIFO alternatively. When data from first half of FIFO is transferring to the SD3.0 / SDIO3.0 / eMMC4.41 card, the second half of FIFO will be filled and vice versa. The two halves of the FIFO’s are alternatively used to store data which will give maximum throughput. During a read transaction (data transferred from SD3.0 / SDIO3.0 / eMMC4.41 card to ARM Processor), the data from SD3.0 / SDIO3.0 / eMMC4.41 card will be written in to the two halves of the FIFO alternatively. When data from one half of the FIFO is transferring to the ARM Processor, the second half of the FIFO will be filled and vice versa and thereby the throughput will be maximum. If the Host controller cannot accept any data from SD3.0 / SDIO3.0 / eMMC4.41 card, then it will issue read wait to stop the data transfer from card or by stopping the clock.

C. *Command and Data Control logic:*
The DAT [0-7] control logic block transmits data on the data lines during write transaction and receives data from the data lines during read transaction. The DAT [0-7] control logic block transmits data in the data lines on posedge and negedge of the SD CLOCK during DDR mode of operation. The DATA [0-7] receiver block receives/ samples the data on the data lines in both posedge and negedge of the SD CLOCK during DDR mode of operation. The Command control logic block sends the command on the cmd line and receives the response coming from the SD3.0 / SDIO3.0 / eMMC4.41 card.

III. VERIFICATION ENVIRONMENT

The Design Under Test (DUT) i.e. SD Host Controller is verified using BFMs (Bus Functional Model) designed in Verilog HDL. The AHB master and AHB target models assist in generating AHB transactions along with arbiter. They emulate the function of an ARM processor. The organization of the verification environment is shown in above figure 3.

A. *AHB Master/Slave BFM & Arbiter:*
The Master BFM will drive the system BUS (AHB or AXI or OCP etc) interface signals. This is the processor read/write interface. The slave BFM is added to respond to the DMA master (Ex: ADMA descriptor read and DMA write/ read transfer). This is the DUT DMA interface which transfers data to and from system memory without the intervention of the processor.

Arbiter block takes care of the bus arbitration for the AHB bus when the core is the master. It takes the requests lines from the master core and gives grant according to the priority.

B. *Bus Monitor & Score Board:*
The bus monitors are used to check for protocol violations and to display information about the traffic.

AHB Bus Monitor:
AHB Bus monitor file will monitor the AHB Bus and store the data into its internal memory array. It will also monitor the contents driven through the BUS for ADMA operation, DMA and PIO mode of transaction.
SD Monitor:
SD Bus monitor file will monitor the SD Bus and store the data into its internal memory array. It

decodes the CMD and DATA lines and fetches the information accordingly.

Score Board:

Score Board is used for data integrity check after data transmission. The expected data and the actual data are fed to the scoreboard for comparison. The test fails if there is data mis-compare or if transmitted data is not equal to the requested transfer.

C. SD Host command definitions (API File):

This file contains all the commands what the host driver will use for the device. The definition of each command shall invoke the system write and read tasks and pass the corresponding arguments.

ADMA descriptor calculation task is used to calculate the ADMA descriptor and program the same in the system slave memory for ADMA2 transaction.

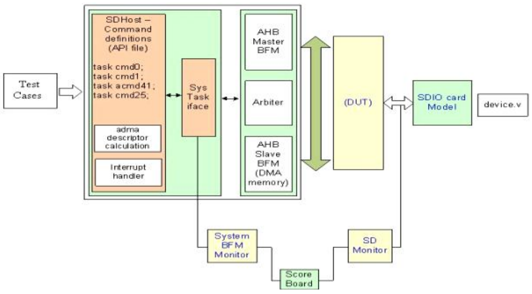


Fig. 3 : Verification Environment for SD Host Controller

Interrupt handler services all the host controller interrupts and will call the data transfer task and error recovery tasks automatically.

D. System Task Interface:

The main purpose of this file is to randomize the system write tasks (i.e., randomly call “sys_write32” or “sys_write16” or “sys_write8” depends on the number of bytes are passed in the “sys_write”). It also supplies the necessary information to the System BFM monitor for data integrity.

E. Test Cases:

The test cases covering different scenarios are written separately and referred with different names. Perl Scripts are written to pick the test case as our wish, compare the files generated during simulations and running regression.

IV. DATA TRANSFER MODES

The SD Host Controller supports three modes to transfer Data between System and the card. The three modes include Non-DMA, DMA and ADMA mode. In Non-DMA mode (also known as "programmed I/O" method) the Host Controller has a data buffer for data transfer. The Host Driver accesses internal buffer through the 32-bit Buffer Data Port register. Internally, the Host Controller maintains a pointer to control the data buffer. The pointer is not directly accessible by the Host Driver. Every time the Buffer Data Port register is accessed, the pointer is incremented depending on amount of data written to.

In DMA mode, the data transfer between system memory and SD card takes place without interruption of CPU execution. DMA shall support both single block and multiple-block transfers but does not support infinite transfers. The result of a DMA transfer shall be the same regardless of the system bus data transfer method. DMA had disadvantage that DMA Interrupt generated at every page boundary disturbs CPU to reprogram the new system address. This DMA algorithm forms a performance bottleneck by interruption at every page boundary.

A new DMA transfer algorithm called ADMA (Advanced DMA) is defined that adopts scatter gather DMA algorithm so that higher data transfer speed is available. The Host Driver can program a list of data transfers between system memory and SD card to the Descriptor Table before executing ADMA. It enables ADMA to operate without interrupting the Host Driver. Furthermore, ADMA can support not only 32-bit system memory addressing but also 64-bit system memory addressing. The 32-bit system memory addressing uses lower 32-bit field of 64-bit address registers.

Block Diagram of ADMA2

Figure 4 shows block diagram of ADMA2. The Descriptor Table is created in system memory by the Host Driver. 32-bit Address Descriptor Table is used for the system with 32-bit addressing and 64-bit Address Descriptor Table is used for the system with 64-bit addressing. Each descriptor line (one executable unit) consists with address, length and attribute field. The attribute specifies operation of the descriptor line. ADMA2 includes SDMA, State Machine and Registers circuits. ADMA2 does not use 32-bit SDMA System Address Register (offset 0) but uses the 64-bit Advanced DMA System Address register (offset 058h) for descriptor pointer. Writing Command register triggers off ADMA2 transfer. ADMA2 fetches one descriptor line and executes it. This procedure is repeated until end of descriptor is found.

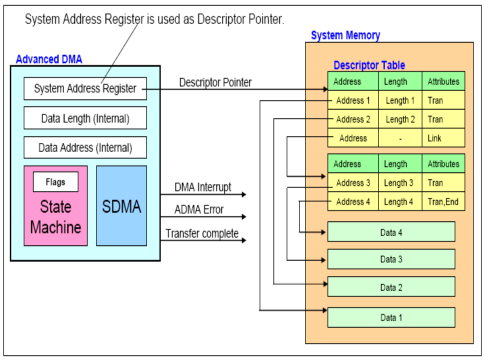
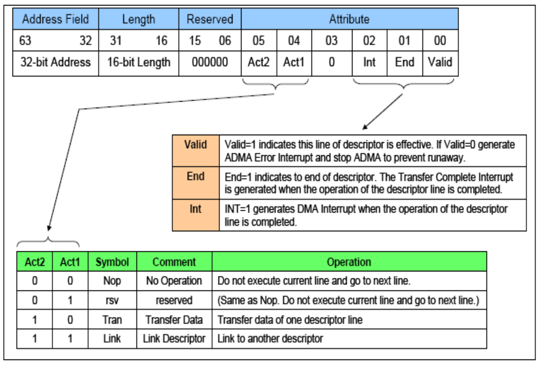


Fig. 4 : Block Diagram of ADMA2

Descriptor Table

Figure 5 shows the definition of 32-bit Address Descriptor Table. One descriptor line consumes 64-bit (8-byte) memory space. Attribute is used to control descriptor. 3 action symbols are specified. "Nop" operation skips current descriptor line and fetches next one. "Tran" operation transfers data designated by address and length field. "Link" operation is used to connect separated two descriptors. The address field of link points to next Descriptor Table.



Address Field		Length		Reserved		Attribute					
63	32	31	16	15	06	05	04	03	02	01	00
32-bit Address		16-bit Length		000000		Act2	Act1	0	Int	End	Valid

Valid	Valid=1 indicates this line of descriptor is effective. If Valid=0 generate ADMA Error Interrupt and stop ADMA to prevent runaway.
End	End=1 indicates to end of descriptor. The Transfer Complete Interrupt is generated when the operation of the descriptor line is completed.
Int	INT=1 generates DMA Interrupt when the operation of the descriptor line is completed.

Act2	Act1	Symbol	Comment	Operation
0	0	Nop	No Operation	Do not execute current line and go to next line.
0	1	rsv	reserved	(Same as Nop. Do not execute current line and go to next line.)
1	0	Tran	Transfer Data	Transfer data of one descriptor line
1	1	Link	Link Descriptor	Link to another descriptor

Fig. 5 : 32-bit Address Descriptor Table

ADMA2 States

Figure 6 shows state diagram of ADMA2. 4 states are defined; Fetch Descriptor state, Change Address state, Transfer Data state, and Stop ADMA state.

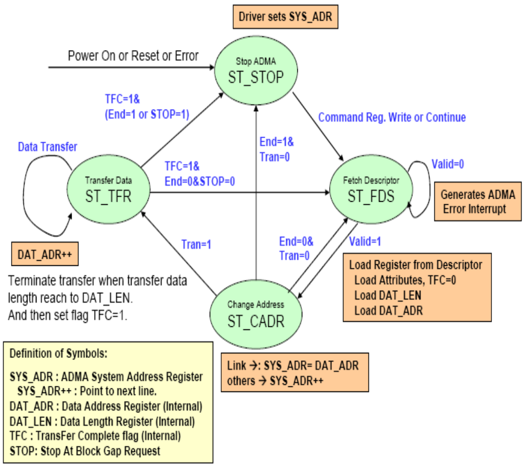

Driver sets SYS_ADR
Power On or Reset or Error
Stop ADMA
ST_STOP
TFC=1&
(End=1 or STOP=1)
Command Reg. Write or Continue
Data Transfer
End=1&
Tran=0
Transfer Data
ST_TFR
TFC=1&
End=0&STOP=0
Fetch Descriptor
ST_FDS
Valid=0
Generates ADMA Error Interrupt
DAT_ADR++
Tran=1
End=0&
Tran=0
Valid=1
Load Register from Descriptor
Load Attributes, TFC=0
Load DAT_LEN
Load DAT_ADR
Terminate transfer when transfer data length reach to DAT_LEN.
And then set flag TFC=1.
Change Address
ST_CADR
Definition of Symbols:
SYS_ADR : ADMA System Address Register
SYS_ADR++ : Point to next line.
DAT_ADR : Data Address Register (Internal)
DAT_LEN : Data Length Register (Internal)
TFC : TransFer Complete flag (Internal)
STOP: Stop At Block Gap Request
Link →: SYS_ADR= DAT_ADR
others → SYS_ADR++

Fig. 6 : State Diagram of ADMA2

Operation of each state is explained in Table-1 below

State Name	Operation
ST_FDS (Fetch Descriptor)	ADMA2 fetches a descriptor line and set parameters in internal registers. Next go to ST_CADR state.
ST_CADR (Change Address)	Link operation loads another Descriptor address to *ADMA System Address* register. In other operations, *ADMA System Address* register is incremented to point next descriptor line. If End=0, go to ST_TFR state. ADMA2 shall not be stopped at this state even if some errors occur.
ST_TFR (Transfer Data)	Data transfer of one descriptor line is executed between system memory and SD card. If data transfer continues (End=0) go to ST_FDS state. If data transfer completes, go to ST_STOP state.
ST_STOP (Stop DMA)	ADMA2 stays in this state in following cases: (1) After Power on reset or software reset. (2) All descriptor data transfers are completed If a new ADMA2 operation is started by writing Command register, go to ST_FDS state.

V. RESULTS

Simulation report:

The Test cases were developed for verifying the different modes of Data Transfer. The three modes include non-DMA, DMA and ADMA mode. The test case is written for different block size and block count and waveforms are verified.

The figure shows the snap shot of verification results.

1. Simple non DMA data transfer operation:

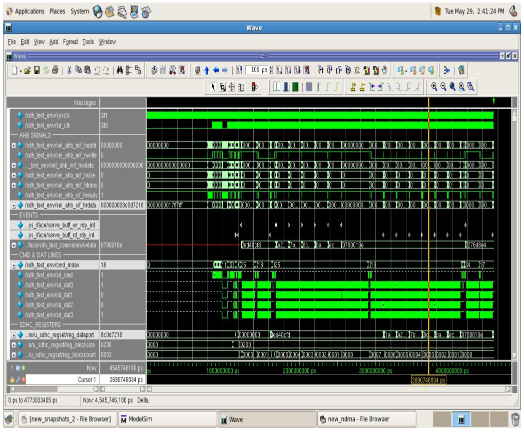


Fig. 7: Waveform of Simple Non-DMA transfer

2. Simple DMA mode Data transfer operation:

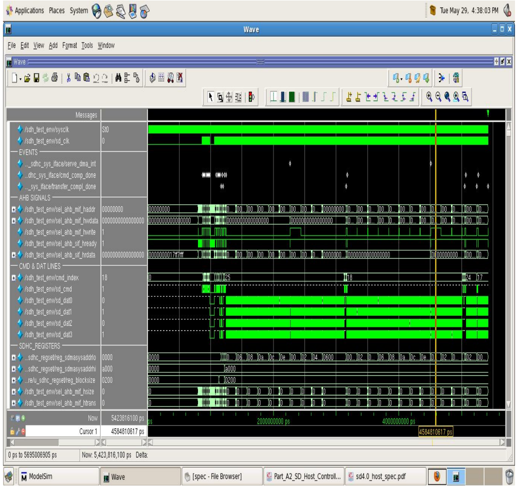


Fig. 8: Waveform of Simple DMA mode transfer

3. Simple ADMA mode Data transfer operation:

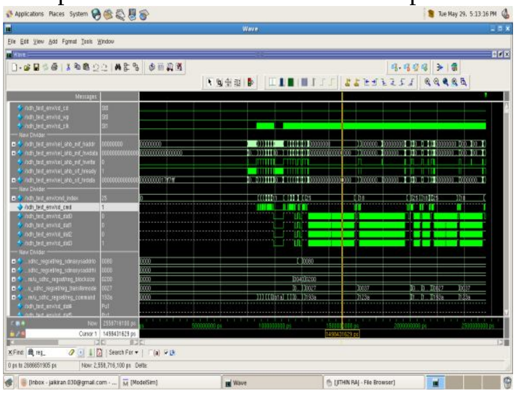


Fig. 9 : Waveform of Simple ADMA mode transfer

VI. CONCLUSIONS AND FUTURE WORK

We developed the verification environment and verified the Secure Digital Host Controller using BFMs (Bus Functional Model) designed in Verilog HDL. Test Cases covering different scenarios are written in Verilog HDL to verify various functionalities of Host controller. The functionalities include DMA, ADMA and Non DMA mode of Data transfer. Future improvement in the SD Host Controller is to add UHS-II mode which can provide bus speed up to 312 Mbps.

REFERENCES

[1] Technical Committee SD Association. (2011): SD Specifications Part A2 SD Host Controller Simplified - Specification Version 3.00. Retrieved February 25, 2011 from SD website: http://www.sdcard.org.

[2] Technical committee SD association. (2011): SD Specifications Part 1Physical Layer Simplified Specification Version 3.01. Retrieved February 18, 2011 from SD website: http://www.sdcard.org.

[3] AMBA Advanced High Performance Bus (AHB) Protocol specification. http://www.arm.com/

[4] Chuan-Sheng Lin and Lan-Rong Dung, “A NAND Flash Memory Controller for SD/MMC Flash Memory Card”. IEEE Transactions on Magnetics, vol.43, No.2, February2007.

[5] P. Rashinkar, P. Paterson, Leena singh, “System-on-a-chip verification methodology and techniques,” Kluwer Academic Publishers, 2001.