Implementation of 4-Bit Data Transmission for Accessing SD Card with FPGA Embedded Soft Processor

Gul Munir Ujjan
School of Electrical & Electronic Engineering, Engineering Campus, Universiti Sains Malaysia, 14300, Penang, Malaysia
Computer & Information Systems Engineering Department, NED University of Engineering & Technology, University road Karachi - 75270, Sindh, Pakistan
gulmunir@neduet.edu.pk

Abdul Malik
School of Electrical & Electronic Engineering, Engineering Campus, Universiti Sains Malaysia, 14300, Penang, Malaysia
abdulmalik@usm.my

Shakil Ahmed
Department of Computer Engineering, Sir Syed University of Engineering and Technology Karachi, Pakistan
atshakil@yahoo.com

Mohd Zaid Abdullah
School of Electrical & Electronic Engineering, Engineering Campus, Universiti Sains Malaysia, 14300, Penang, Malaysia
+60 4 599 6030
mza@usm.my

ABSTRACT
Secure Digital (SD) cards being removable, non-volatile and flash memory in nature, are highly preferred for use in FPGA based systems as secondary storage. Currently most FPGA developers prefer 1-bit SD mode data transmission as a mean to access the SD card due to the complexity of 4-bit serial design. Hence, the speed is significantly compromised. This paper discusses the design and implementation of a firmware for 4-bit SD mode data transmission. A simple hardware application based on switches, buttons, memory and SD card interface is used to illustrate the functionality of proposed firmware, particularly the generation of the required control signal for data access. Meanwhile the Verilog HDL is used for hardware design while the software control is coded in C language. The system is implemented on Altera DE-4 board with FPGA Stratix IV GX EP4SGX230 and 32-bit NIOS-II embedded soft processor. In terms of data throughput the new firmware is 80 % much faster compared to a standard 1-bit data transmission when accessing a single block of data.

CCS Concepts
• **Hardware→Communication Hardware**, I**nterfaces and Storage.**

Keywords
4-bit/1-bit SD mode; FPGA; NIOS-II processor; Altera DE-4 board; Stratix IV.

ICIIT '19, February 20–23, 2019, Da Nang, Viet Nam

ACM ISBN 978-1-4503-6633-5/19/02…$15.00
DOI: https://doi.org/10.1145/3321454.3321475

1. INTRODUCTION
Field Programmable Gate Arrays (FPGA) due to their reconfigurable nature [1][2], high computational power[3], supporting real time processing [4] and parallelism [2][3][4]are increasingly being incorporated into ASIC and SOC designs. Like any processing unit, FPGAs also rely on memory systems that may be either only short-term primary memory or an optional long-term secondary memory. Currently the on-chip or on-board memories available from vendors' FPGA or development boards are usually smaller in sizes, thereby leaving less flexibility for designs which rely on high computational power and parallelism together with greater space to store their secondary nature data. Therefore, the hardware designers are always considerate about secondary memory and its available sizes before actually considering the design and implementation of the data intensive systems. Such systems may include the HD (High definition) image and video processing systems, the speech recognition and processing systems, the ANN (Artificial Neural Network) based learning systems, the medical imaging systems to name a few.

SD cards are removable, non-volatile, shock resistant flash-based storage devices and favored as secondary storage because of their smaller sizes, lower power consumption and lower cost [5][6][7]. They support multiple write characteristics and easy to handle as compared to bulky mechanical hard disk drives [7][8].Nowadays SD cards are very popular among small consumer devices like PDAs, digital cameras, hand phones and other portable devices [9]. Similarly, FPGA based systems regularly use SD cards for long-term storage, as their native or on-chip and on-board memories are apparently very small in sizes and mostly volatile in nature. Like any other storage types, SD cards also use file system which is responsible for organizing the files and folders on it. SD cards of size 2GB or less, use File Allocation Table (FAT) FAT-12 or FAT-16 as the file system and those of size 4GB or above normally use FAT-32 file system [10].Currently the SD card interface available with the FPGA boards already have an option of 4-bit design upgrading in addition to a default 1-bit SD and SPI modes (which allows only a subset of SD commands). Even then most FPGA designers prefer clinging to 1-bit SD mode because of the difficulty in designing a firmware for 4-bit data transmission.

Such a difficulty arises principally from complex timings structure when transmitting at higher frequency as well as designing a mandatory 16-bit error correction [10]. This paper presents a firmware for utilizing SD card as a secondary storage based on 4-bit read/write mode. A 1-bit SD mode is implemented for analysis and comparative understanding of data transfer in SD modes.

This paper is organized as follows. Section 2 discusses the hardware design while basic principles of SD card is explained Section 3. The read and write operations are presented in Section 4, and followed by the software implementation of read and write operations for both SD 4-bits and 1-bit modes in Section 5. Experimental results and discussion are presented in Section 6, and conclusions are drawn at the end in Section 7.

2. SYSTEM HARDWARE

This development work primarily is based on embedded 32-bit NIOS-II FPGA soft processor, comprising of some basic Parallel Input/Output (PIO), DDR2 (Double-Data Rate) RAM and on chip memory. Specifically the system is using the SD card interface, and it is also provided with 50MHz system clock and reset interfaces. The system can either read or write data on SD card through push of a button. DDR2 can be read or written in similar manner. Here DDR2 is just provided as a prototype for live data storage, for any system to be implemented later. That live data which can be generated by any sensor, HD camera etc. initially be stored on DDR2, can then be stored on SD card also for long-term use. This system comes with built-in software for selecting DDR2, or SD card or a temporary program variable as a data source. Similarly any of these sources can otherwise exclusively act as a destination of data. In other words this system creates an easy data interface between SD card, DDR2 and program variables.

2.1 System Specifications

The system is implemented on Altera Stratix IV GX EP4SGX230 FPGA integrated with DE-4 (Development and Education board). Altera digital designing platform Quartus-II is used and then Verilog is selected as HDL (Hardware Description Language). The software programs and subroutines of SD card commands, drivers and FAT-32 file system are all implemented in C language using NIOS-II SBT (Software Build Tools) for Eclipse. The working programs for file system and specific drivers are used from Altera demo projects but modification and addition is done to work for this hardware system especially for 4-bit SD mode.

2.2 System Design

Hardware design of this system is done using one of the Quartus-II integrated Platform Designer tool. The design showing various hardware components, IPs (Intellectual Properties) and their interconnections is given in Figure 1.

New system design can be generated by simply following the pick-place-interconnect steps, and applying some custom changes primitive to DE-4 board.

Once the design is complete, the system is generated by selecting Verilog as the target HDL as in this case. The designer tool then gives a top-level Verilog module hierarchically connected with several required Altera IP modules through their Input/Output ports. After the successful system generation, a new top-level Verilog module is added into a designer generated module. This enabled further custom features be applied, like assigning oscillator clock and reset inputs, turning on/off LEDs, and instantiating the core design modules etc.

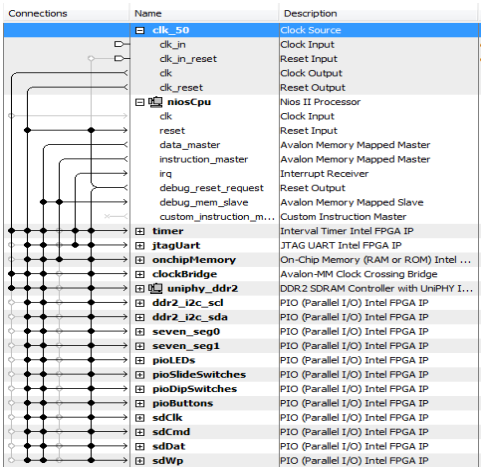

Figure 1. System hardware design.

Following the successful completion of the above tasks, the pins are appropriately assigned and followed by setting proper timing and delay constraints using Time Quest timing analyzer tool. Only then the system is ready to be fully synthesized and compiled. Once full compilation is successful, the generated program binary file is loaded into FPGA using Quartus-II integrated programmer tool through USB blaster cable. The hardware is now ready to work, testing and emulating the interfacing of various hardware components. As the system needs to establish communication with components like DDR2 and SD card using a 32-bit soft processor NIOS-II, therefore a software which can run on NIOS-II is needed. Doing this requires some basics understanding of SD card which is explored in the next section.

3. SD CARD

3.1 Bus Protocols

The SD card uses two different set of bus protocols which are SPI (Serial Parallel Interface) and SD protocols.SD protocol being faster is the default protocol to work on a start-up. However the SD card can also be configured during first reset cycle to work with a slower SPI protocol by asserting a negative logic on CS signal [10].

With SPI protocol the card operates in just 1-bit SPI mode using the most generic SPI bus interface. It supports a subset of SD commands only. CRC (Cyclic Redundancy Check), which is a method of detecting errors in transmitted data, is also optional to be used with SPI mode [10]. This mode being straightforward and simple is highly preferred for use with devices when data transfer throughput is not an issue.

3.2 SD Bus Protocol

Since the main focus of this paper is SD protocol then the discussion in this section and the subsequent sections following this is limited to this topic only. Table 1 summarizes the pin assignments of a typical SD card.
SD protocol supports two different modes which are SD 1-bit and SD 4-bit modes where the card operates in a clock serial mode with widths of 1-bit and 4-bits respectively. An application specific command ACMD6 is used during card initialization phase to switch the bus width from 1-bit to 4-bit and vice versa [10]. Both of these modes are used for data transfers in bulk quantity. The SD card comprises of 9 pins which are all used to connect with the host for data access. As listed in Table 1, it

consists of 4 pins for bidirectional data, 3 for power supply, 1 clock line and 1 bidirectional command line for sending commands and receiving the card responses.

Table 1. Pin-out descriptions for SD card in SD modes [10]

Pin#	Name	Type	SD Description
1	CD/DAT3	Bidirection	Card detection/ SPI mode selection/ data line [Bit 3]
2	CMD	Bidirection	Command/Response
3	V_{SS1}	Supply	Supply voltage ground
4	V_{DD}	Supply	Supply voltage
5	CLK	Input	Clock
6	V_{SS2}	Supply	Supply voltage ground
7	DAT0	Bidirection	Data line [Bit 0]
8	DAT1	Bidirection	Data line [Bit 1]
9	DAT2	Bidirection	Data line [Bit 2]

SD 1-bit mode uses DAT0 only for bidirectional data transfer. In contrast SD 4-bit mode uses all four data lines, i.e.DAT0-DAT3 for the same purpose. For this reason 4-bit mode is expected to work much faster almost 4x speed than the 1-bit counterpart [11].

3.3 SD Data Packet Formats

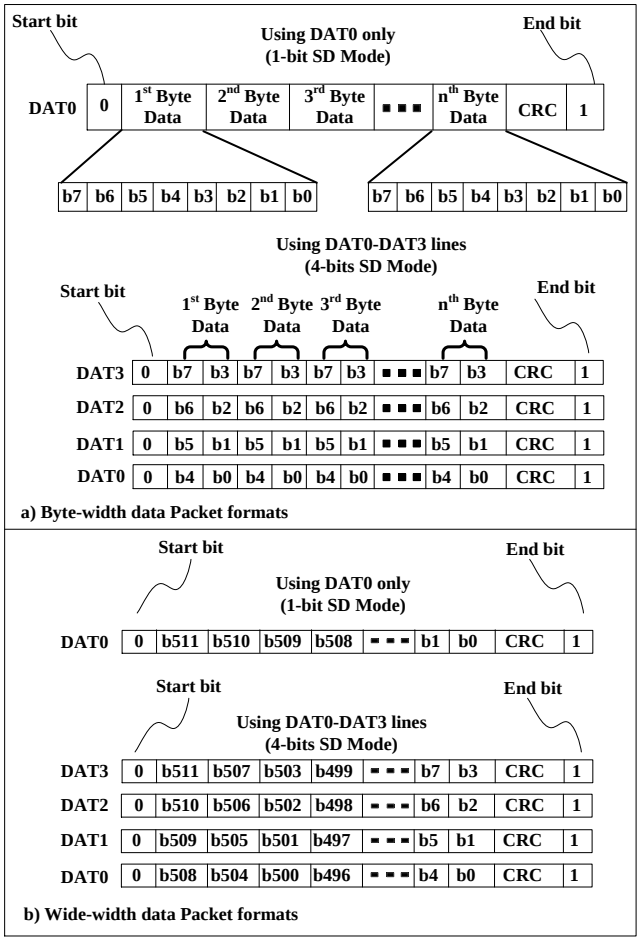

Figure 2. SD protocol data packet formats [10].

In either read or write or status or response operations the data transfer is done in blocks. The block size of most high capacity SD cards (SDHC) is usually 512 bytes. Software implemented in this system is also using 512 bytes as block size for data read and write operations. There are two types of Data packet formats for the SD card in SD modes. They are (i) Byte-width and (ii) Wide-width data formats, as given in Figure 2.

Beginning with the least significant byte first, in byte-width data format, the individual bits are transferred as most significant bit first and the least significant bit as the last. While in the wide-width data format, the whole data is shifted starting from the most significant bit. In both cases, each transfer must always begin with 0 as the starting bit and 1 as the ending bit. In 4-bit SD mode, all individual DAT lines will carry 128 bytes summing up as a block of 512 bytes as clearly shown in Figure 2.

Unlike SPI both SD modes rely on 16-bit CRC error protection for data transfers. The data on DAT0 must be suffixed with a 2-byte CRC code before the transfer end bit in SD 1-bit mode. In 4-bit SD mode, all DAT0-DAT3 lines must also be suffixed by 2-byte CRC code before the transfer end bits in all DAT lines. And in the second case, the 16-bit CRC code must be calculated and tested for all DAT lines separately [10].

3.4 SD Command Format

The control signals or SD commands are needed to control and give instructions to the SD card for any required function. Here CMD line is used for issuing commands and also for receiving response as a feedback against the issued commands in SD modes. All commands in the command set are presented as a packed frame of 6 bytes with specific format structure as shown in Figure 3. Transmission bit is 1 in case of command and 0 for feedback response. A 6-bit unique command number instructs the card what to do and an optional 32-bit argument defines necessary parameters. In this case CRC7 is a 7-bit CRC code for command integrity verification.

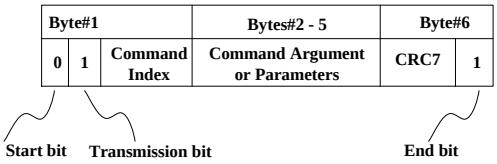

Figure 3. SD Command Frame structure [10].

As 6 bits are reserved for command index therefore SD card can support 64 commands. Interested readers can refer to [10] for detailed description of each SD command. In this paper the single block read (CMD17) and write (CMD24) commands are discussed and hence implemented in this paper.

4. SD CARD READ / WRITE COMMANDS

SD commands CMD17 and CMD24 are used for reading and writing a single block of 512 bytes respectively. Also the card allows CMD18 and CMD25 for multiple block reads and writes operations. Once used these read/write operations need to be terminated by activating CMD12 separately, indicating the end of data transmission. Both CMD17 and CMD24 are 48-bit long commands and in compliance with the format in Figure 3. These commands are activated by the host using CMD line with transmission bit as 1 meaning from host to card.

In response the SD card reset the transmission bit to 0. The response defines whether the command is accepted or there is any

error due to CRC, illegal parameters, undefined command etc. For actual data read or write to occur, 1-bit SD mode uses DAT0 whereas DAT0-DAT3 lines are used for SD 4-bit mode. Here all read/write bytes are formatted according to sequence shown in Figure 2. Individually these operations are represented in a block diagram as shown in Figure 4 (a) and b) for read and write operation respectively.

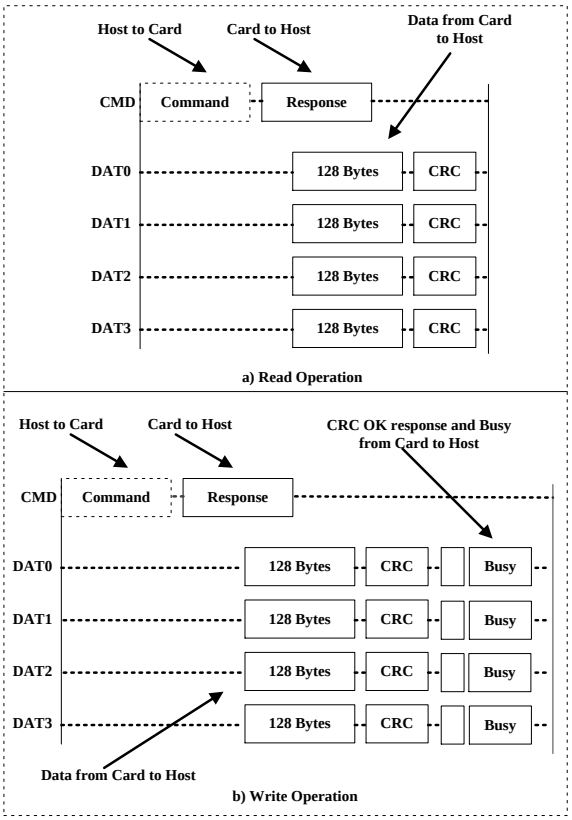

Figure 4. Single block 4-bit mode read/ write operations.

A 16-bit CRC code follows the read or write data block read for data integrity checking at either sides. In case of data write command the CRC calculations are performed by the host(on DAT0 only for 1-bit SD mode or DAT0-DAT3 lines individually for SD 4-bit mode). The results will then be communicated to the SD card. A few clock cycles after receiving the CRC, the card will generate a 1-byte CRC ok response on DAT0 line only (even in case of DAT0-DAT3 are used). This 1-byte response is decoded as in Figure 5 where 'x' represent don't care.

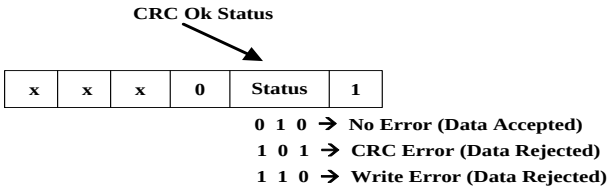

Figure 5. CRC Ok response for data [10].

The card goes busy in physically dumping the data on its addressed sectors if CRC response is ok.

5. SOFTWARE IMPLEMENTATION

5.1 SD Card Mount and Initialize

Referring to Figure 6 and Figure 7, both the read and write operations begin with SD card mounting and initializing processes. While mounting, the software checks card type and capacity whether it is low or high capacity card, and whether it is standard SD card or micro SD card etc. It checks the file system like FAT-32 or FAT-16, then browse the directories and files. After the file is opened or new one created with a given name, then the software temporary stores the file contexts like name, type, size, date and time, logical cluster address, etc. on SD card. Once mounting is successful, the card goes through initialization steps like peeking and poking the internal status and control registers, setting clock, checking the operating voltage levels which ranges from 2.7V to 3.6V. ACMD6 may also be used in initialization process in order to set the bus width to 4-bit SD mode as explained in Section 2. Overall the initialization process makes sure that the card is fully functional and capable to respond to any general commands.

5.2 Read Operation (CMD17)

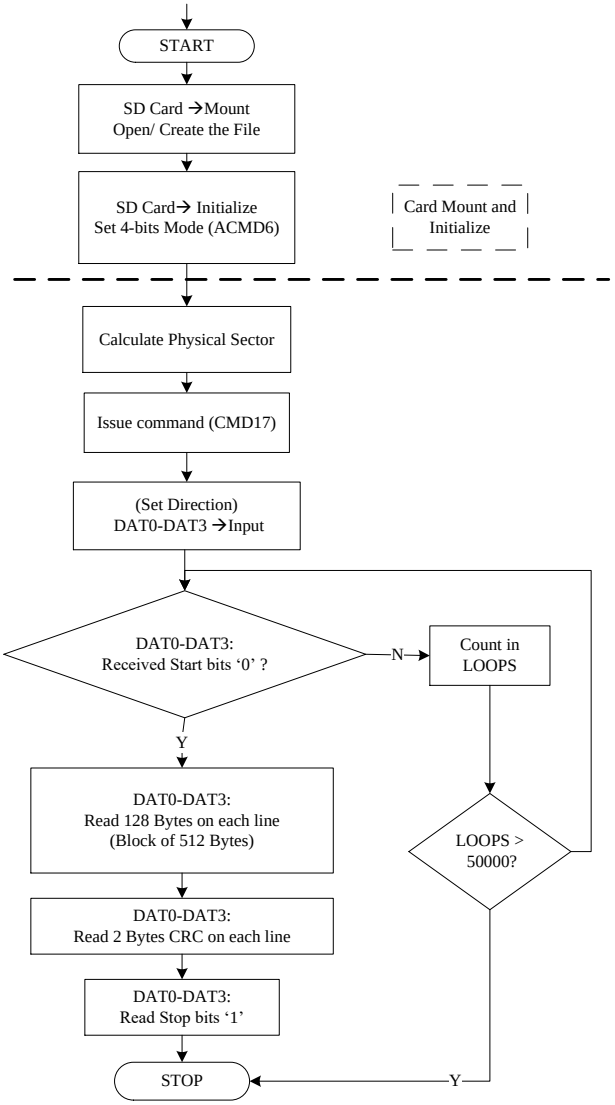

Figure 6. Read operation (CMD17) flow diagram.

While the file context is already stored in temporary memory, the software calculates physical sector address from its saved logical cluster address in order to physically access the file area on SD card. Then a single block read command CMD17 (refer to Figure 7) is issued followed by setting direction of all data lines to input (default direction is also input).

If no error in command, the card will then start transmitting data by placing '0' on each DAT lines. Unless start bits are not received, the program loops back for maximum 50000 iterations until the card is ready. Once start bits are received, the host will read a block of 512 bytes of data with 128 bytes on each DAT lines. The positioning of bits and bytes are formatted according to Figure 2. At the end of the operation, all DAT lines will receive a 2-bytes CRC code for data received by individual lines followed by stop bits '1'.

5.3 Write Operation (CMD24)

Once the physical sector is calculated, the software issues the command CMD24 as in Figure 7. Explicitly the direction of DAT lines will be set to output. Similar to CMD17, the start and stop bits will also be transferred in write operation but this time from host to the card. After SD card receives 512 bytes of data (128 bytes on each line) and individual lines' 16-bit CRC, the card will respond with 1-byte CRC ok response (refer to Figure 5).Only after that, the card goes busy in physically writing the data at the addressed sector. It remains in busy state unless write operation is complete.

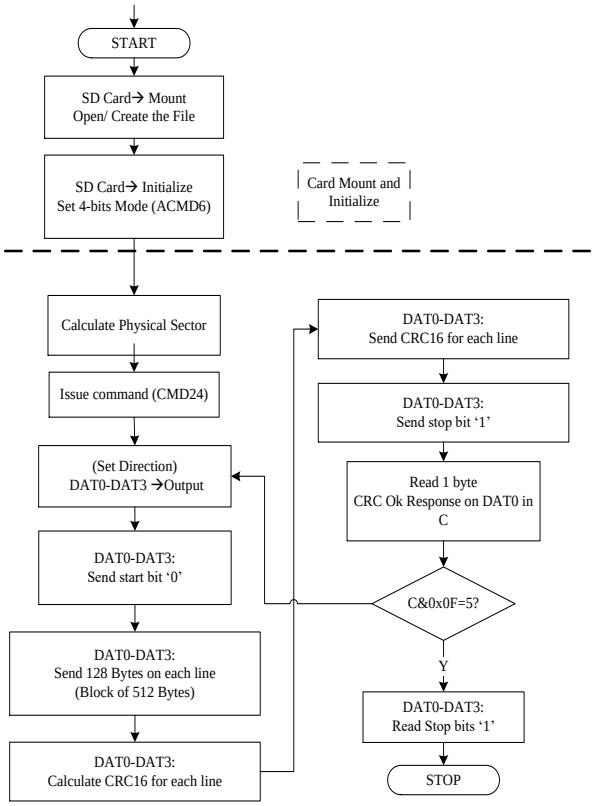

Figure 7. Write operation (CMD24) flow diagram.

6. RESULTS AND DISCUSSION

As already mentioned in Section 2 and Section 3, the proposed firmware is capable of establishing an interface between DDR2 and SD card. This is useful for any application which generates data, and needs long-term storage like SD card. Running on NIOS-II soft processor the software can read data from DDR2 and write to the SD card or vice versa. Both read and write operations can be performed either through 1-bit or 4-bit SD mode. However only a read operation is included in the discussion in order to demonstrate the capability and capacity of the firmware. The snapshots which illustrate various data transfer options are included and discussed in the following sub-sections.

6.1 Data Transfer

6.1.1 Reading from DDR2 memory

```
Reading from DDR
07d0:e16ff14e 07d1:00a11080 07d2:20e330c2 07d3:40255004
07d4:60677046 07d5:939883b9 07d6:b3daa3fb 07d7:d31cc33d
07d8:f35ee37f 07d9:129002b1 07da:32d222f3 07db:52144235
07dc:72566277 07dd:a5cbb5ea 07de:858995a8 07df:e54ff56e
07e0:c50dd52c 07e1:24c334e2 07e2:048114a0 07e3:64477466
07e4:44055424 07e5:b7faa7db 07e6:97b88799 07e7:f77ee75f
07e8:d73cc71d 07e9:36f226d3 07ea:16b00691 07eb:76766657
07ec:56344615 07ed:c96dd94c 07ee:e92ff90e 07ef:89e999c8
07f0:a9abb98a 07f1:48655844 07f2:68277806 07f3:08e118c0
07f4:28a33882 07f5:db5ccb7d 07f6:fb1eeb3f 07f7:9bd88bf9
07f8:bb9aabbb 07f9:5a544a75 07fa:7a166a37 07fb:1ad00af1
07fc:3a922ab3 07fd:ed0ffd2e 07fe:cd4ddd6c 07ff:ad8bbdaa
0800:8dc99de8 0801:6c077c26 0802:4c455c64 0803:2c833ca2
0804:0cc11ce0 0805:ff3eef1f 0806:df7ccf5d 0807:bfbaaf9b
0808:9ff88fd9 0809:7e366e17 080a:5e744e55 080b:3eb22e93
080c:1ef00ed1 080d:1b120900 080e:3f362d24 080f:535a4148

DDR2 read operation successful, size=1024 bytes, 0.001 sec
```

Figure 8. Example of DDR2 read operation.

A simple read operation for DDR2 is given in Figure 8. With the help of NIOS-II processor, the system accesses 1KB of data in just 1 ms. The data read is displayed in address:word format with 32-bit word size and hexadecimal data type. Here DDR2 write operation is ignored.

6.1.2 Reading from SD Card

```
Reading (only) from SD Card
4-BITS SD MODE
Root Directory Item Count:3
READING...

In addition to the SD Memory Card, there is the SD I/O (
is defined in a separate specification named: SDIO Card
the SD Association. The SDIO Specification defines an SD
n various I/O units and an SD Host. The SDIO card may co
as its I/O functionality. The Memory portion of SDIO car
 SD Memory Card specification

Time elapsed reading 510 bytes from sd card: 0.003 sec
```

a) 4-bits SD Mode

```
Reading (only) from SD Card
1-BIT SD MODE
Root Directory Item Count:3
READING...

In addition to the SD Memory Card, there is the SD I/O (
is defined in a separate specification named: SDIO Card
the SD Association. The SDIO Specification defines an SD
n various I/O units and an SD Host. The SDIO card may co
as its I/O functionality. The Memory portion of SDIO car
 SD Memory Card specification

Time elapsed reading 510 bytes from sd card: 0.005 sec
```

b) 1-bit SD Mode

Figure 9. Example of SD card read operation.

Figure 9shows the snapshots for the single block read operations using both SD modes. As printed at the bottom of this snapshot, the SD 1-bit mode takes 5 ms while 4-bit mode reads the same 512 bytes in 3 ms with difference in data throughput of 69 bytes or 67 % faster.

6.1.3 Writing to SD Card

Results of a single block write operations using both the SD modes are given in Figure 10. It can be seen that the SD 4-bit mode takes approximately 5 ms to write as compared to 9 ms for 1-bit mode, resulting in a data throughput difference of approximately 46 bytes which is approximately 80% faster.

Since only one block is involved in reading or writing, therefore, there is no significant difference in terms of data throughput. When accessing several blocks by using multi-block read/ write operations, the speed can significantly be increased especially for 4-bit SD mode. The throughput of data for 4-bit mode may also improve depending on other factors like adding separate processor or logic for calculation of 16-bit CRC, and increasing the size of DDR or main memory. The performance improvement will out weight the cost of additional use of main memory. Therefore adding up extra logics and additional hardware support will improve the speed of read and write operations significantly.

```
Writing to SD Card
4-BITS SD MODE
Root Directory Item Count:3
WRITING...
Response Code is: 0x05
SD Memory Card is a memory card that is specifically de
ements inherent in newly emerging audio and video consu
ion mechanism that complies with the security of the SD
 SD Card security system uses mutual authentication and
Response Code is: 0x05

Time elapsed Writing 512 bytes to SD card: 0.005 sec
```

a) 4-bits SD Mode

```
Writing to SD Card
1-BIT SD MODE
Root Directory Item Count:3
WRITING...
Response code is: 0x05
SD Memory Card is a memory card that is specifically de
ements inherent in newly emerging audio and video consu
ion mechanism that complies with the security of the SD
 SD Card security system uses mutual authentication and
Response code is: 0x05

Time elapsed Writing 512 bytes to SD card: 0.009 sec
```

b) 1-bit SD Mode

Figure 10. Example of SD card write operation.

7. CONCLUSION

A primary hardware system based on FPGA and embedded NIOS-II processor for accessing SD card in 4-bit SD mode is implemented. The software which runs on NIOS-II processor is written in C language using NIOS-II eclipse platform. Using FAT-32 file system, single block read and write commands are implemented and tested. For comparison, these commands are also implemented for SD 1-bit mode. The resulting throughput of read operation with SD-4 bit mode proves it is nearly 67 % faster than SD 1-bit mode. Similarly, for write operation in which SD 4-bit mode is 80 % faster compared to SD 1-bit mode. Throughput can further be improved using multi-block read and write operations and implementing fast mechanism for calculating 16-bit CRC write operation. This requires extra hardware resources and hence additional costs. The proposed firmware will benefit systems which handle big data transfer like real-time video capturing.

8. REFERENCES

[1] Ke Wang. 2009. An encrypt and decrypt algorithm implementation on FPGAs. *IEEE. Fifth International Conference on Semantics, Knowledge and Grid*, pp.298-301.

[2] Ami J. Shukla, Prof. Vibha Patel, Prof. Nagendra Gajjar. 2015.Implementation of Edge Detection Algorithms in Real Time on FPGA. *IEEE, 5th Nirma University International Conference on Engineering (NUiCONE).*

[3] Zhenlin LU, Jingjiao LI, Yao Zhang. 2010. The Reading/Writing SD Card System Based on FPGA. *2010 First International Conference on Pervasive Computing Signal Processing and Applications (PCSPA)*, pp. 419-422.

[4] Manish Kumar Birla. 2006. FPGA Based Reconfigurable Platform for Complex Image Processing. *IEEE, 6th International Conference on EIT*, pp.204-209.

[5] Altera Corporation. 2010. Memory System Design. *Chapter 7, Embedded Design Handbook*, pp.7-1-7-18, February 2010.

[6] Guo J., Liu D. 2012. Design of a Smart Meter Recorder with Mass Storage Based on DL/T645-2007 Protocol. *Internet of Things. Communications in Computer and Information Science*, vol 312. Springer, Berlin, Heidelberg, pp 660-666.

[7] Gay W.W. 2014. SD Card Storage. *Raspberry Pi Hardware Reference. Apress*, Berkeley, CA, pp. 81-88.

[8] Elkeelany, O. Todakar, V.S. 2011. Data concentration and archival to SD card via hardware description language. *GLOBECOM Workshops (GC Wkshps)*, 2011 IEEE, pp. 625-630.

[9] Yansi Yang. Yingyun Yang. Lipi Niu. Huabing Wang. Bo Liu. 2011. Hardware system design of SD card reader and image processor on FPGA. *Information and Automation (ICIA), 2011 IEEE International Conference on*, pp. 577-580.

[10] SD Card Association. 2006. SD Specifications Part-1 Physical Layer Simplified Specification Version 2.00.

[11] SanDisk Corporation. 2003. SanDisk Secure Digital Card. *Product Manual. Version 1.9.* document no. 80-13-00169.