SD CARD INTERFACE USING FPGA FOR MULTIMEDIA APPLICATIONS

Sakthekannan M S, Udaya Krishnan M, Yogavignes B M

*Electronics and Communication Engineering, Saveetha Engineering College, Chennai.*

[*srivisakthi@gmail.com*](mailto:srivisakthi@gmail.com)

[*udayakrishnan.ece.sec@gmail.com*](mailto:udayakrishnan.ece.sec@gmail.com)

*yogavignesbm @gmail.com*

***Abstract—*** **Based on FPGA (Field Programming Gate Array) chips, this application is used for audio, video and image processing. At the same time, applications require more memory in addition to on-chip memory to resolve additional data. Onboard memory can be used to meet the requirements of FPGA systems but cannot be expanded by adding just a few cards. More convenient is SD card (Secure Digitals) is a microcontroller which can read and write to SD card. The main goal is to provide a storage solution for FPGA, a low-cost, removable, non-volatile, flash memory, portable and easy-to-use storage solution for FPGA for storing large files. The hardware design is done in Verilog HDL language and implemented in FPGA. All data access from the SD card is through Verilog, eliminating the need for an on-chip microcontroller or general-purpose processor.** **Spartan 6 (XC6SLX9-3csg324) FPGA is used. The FPGA runs on 5V power supply with a built-in oscillator frequency of 100 MHz A 4GB micro SDHC card (class 6) from Strontium is used in this particular project. FAT32 is formatted by SD card. The ultimate aim of this project is to read a BMP image file from the SD card. FAT32 is formatted by SD card before interfacing. The code for the FAT32 is written to interface the SD card.**

***Keywords—FPGA, Verilog HDL, SD card, 4-bit SD mode, SPI protocol.***

I. INTRODUCTION

Secure Digital (SD) cards are proprietary non-volatile cards. New generation semiconductor-based memory card an electronic device designed primarily for data storage portable or fixed application. Main advantages of SD cards are small, have large storage capacities and fast data bits, price, high energy efficiency, great flexibility, excellent safety. I have an SD card along with a USB memory stick portable storage device most commonly used in digital systems about microprocessors, microcontrollers and Digital Signals Processor (DSP) or Field Programmable Gate Array in various applications such as (FPGA) chips, digital cameras, laptops, mobile phones, embedded systems [1].

The answer to user needs and requirements is formation of an SD Card Association (SDA). Its purpose is to promote and create the SD card standard card type, speed class, electrical interface, and communication protocol. Supports 3 SD cards communication protocol: 1-bit SD mode, 4-bit SD mode and Serial Peripheral Interface (SPI) mode [1].

SD cards (Secure Digital Memory Cards) are flash memory chips based on new generation semiconductors storage device. Large capacity SD card, high speed data transfer speeds, great flexibility, great mobile security. Considering system size, performance, and storage capacity, it's easy to see that opting for an SD card offers great benefits as a storage device for systems based on FPGA chip.

II. LITERATURE REVIEW

Dumitrel Catalin Costache et al (2020) that a FPGA controller that allows you to write to and read from an SD card using the SPI protocol. SPI transfer mode is essential for easy storage system expansion. Adding the number of SD cards requires that each card share the same clock and data signals and different chip select signals. The proposed architecture requires accessing each card in turn but allows for large amounts of memory.

Gul Munir Ujjan et al (2019) presented that a main hardware system based on FPGA and embedded NIOS-II processor is implemented for accessing SD cards in 4-bit SD mode. The NIOS II Eclipse platform is used to write software that runs on the NIOS II processor. The FAT-32 file system was used to implement and test single block read and write commands. These commands are also implemented in SD 1-bit mode for comparison. SD 4-bit read throughput is approximately 67% faster than SD 1-bit read throughput. The same holds true for write operations, with SD 4-bit mode being 80% faster than SD 1-bit mode. Implementing a fast mechanism that uses multi-block read and write operations and computes 16-bit CRC write operations can improve throughput even more. This necessitates additional hardware resources and thus additional costs. The proposed firmware will benefit systems that handle large amounts of data, such as real-time video capture.

Pallavi Polsani et al (2020) presented that serial synchronous communication via communication between master and slave devices is also used for provisioning communication between microcontrollers and many devices added, similar to external analog to digital converters, digital-to-analog converters, and EEPROMs. There are two different protocols: 1) Inter-I2C and 2) SPI. Both the protocols are well designed for communication between integrated circuits for communication with the onboard neighborhood. SPI is the most commonly used protocol for both at low or medium speeds within and between chips data stream transmission. SPI interface protocol with single master and single slave 8-bit data transmission and all-inclusive configuration. SPI design has been verified and implemented using System Verilog. Demonstrates correctness of coverage code and functionality. The entire RTL is written in Verilog for Synthesis, the verification architecture is written in System Verilog. or implementation is done on Spartan 3E.

Jiayi Qiang et al (2020) presented that the SPI bus is a synchronous, full-duplex, low-signal serial interface data bus line, simple protocol, fast transmission speed. Based on these properties, parallel accelerated computing with FPGAs is used to accommodate device expansion and experimentation high-rate environment. This introduces the structure and working principle of SPI the communication bus analyzes its time structure and four modes of operation and uses this state. A machine method that realizes the SPI bus communication function on FPGA is module circuit. The SPI is written in the Verilog hardware description language, in which waveforms are simulated in vivado simulator. Feasibility of state machine after simulation waveform analysis the method is validated.

Omar Elkeelany et al (2011) that a bidirectional hardware-based core for the SD card design was implemented in a reconfigurable FPGA-based data concentrator chip. In total, 5000 blocks can be written in 1 second and the same 5000 blocks can be read from the SD card in 1.051 seconds. Previous work by the authors took 60 seconds for the same block size. This difference is mainly due to the use of software-based solutions and custom finite-state machine designs found in related work. It shows that real-time data rates of 25 Mb/s can be achieved. This letter presents a real-time design for data archiving to SD cards that can be used in remote site data concentrator applications such as SG. Scalability issues such as the impact of large SD cards and SD cards from different manufacturers will be addressed in future work.

Zinlin et al (2010) presented that it gives a quick creation to the applicable records approximately the SD card. Then primarily based totally on FPGA for the SD reminiscence card reader requirements, it designs FPGA-primarily based totally SD card reader device structure with the diverse practical modules for layout and development. The device become practical tests. By the take a look at results, you may see that the device can examine and write information on SD card, appropriate to fulfil the FPGA device wishes for outside garage devices. At the identical time, this worried the SD card reader era that may be carried out to quite a few FPGA-primarily based totally device. Therefore, the usage of this era must additionally have greater capability applications.

III. PRINCIPLE of SD

*A. Transmission Mode*

SD card supports 3 transmission modes: SPI mode (separate serial input and serial output), 1-bit SD mode (separate command and data channels, one-time transfer format) and 4-bit SD mode (with extra pins and some reset pins 4-wide parallel transmission). SD card mode access speed is faster. The 4-bit SD data transfer rate transmission mode reaches 0-100Mbps and data transfer from 1-bit SD mode rate up to 0~25Mbps. SD transfer mode with 4-bits has a high transfer rate, complex structure and timing make the system more complex difficult to develop [1].

*B. SD Bus Protocol*

The principle of SD communication is simple, master-slave mode. In many cases, this mode includes master and multiple slave devices with 3-6 lines. They are CMD (control line), CLK (clock line), DAT0-3 (data line).

Table I. Pin functions of SD mode

**Name Functions**

CLK microcontroller uses this pin to send a clock signal to SD card.

CMD bidirectional pin for information and command transmission between the microcontroller and the SD card.

DAT0-3 four bidirectional pins are used for bulk data transfer between microcontroller and the SD card.

*C. SD File System*

Structure of FAT16 file system on SD card includes 4 parts. Partition Boot Record (PBR), File Allocation Table (FAT), File Directory Table, and Data Section. A partition boot record typically contains four items: BPB (short for BIOS Parameter Record Block), hard drive flag record book, partition boot record code area, end Flag 55AA. PBR has a BIOS parameter set block. Note that some parameters are the most important. Position of FAT, file directory table, and data area (which sector in SD map) can be computed using these parameters.

*D. SD Command Format*

All SD memory card commands are 6 bytes long including 1-byte command code, including 4-byte command parameters and checksum bytes. Even on order without parameters, the host must also send 4 bytes parameters when sending commands to the SD card. In this case, the parameter can be any value and will be set automatically ignored by the SD card.

Table II SD card commands for SPI mode

**Commands Functions**

CMD0 Software been reset

CMD1 Initialization starts

CMD3 Request card send back the RCA address

CMD7 Card entering the state

CMD9 Read CSD register

CMD10 Read CID register

CMD12 Stop read data

CMD16 Change of size in read/write

CMD17 Single block read command

CMD18 Multiple blocks read command

CMD23 Amount of block sent

CMD24 Single block write

CMD25 Multiple blocks write

CMD32 Erase block start

CMD33 Erase end start

CMD38 Erase command

CMD55 ACMD <n> leading command

CMD58 Read OCR

IV. FPGA

The process of transforming the original system concept into an actual FPGA implementation that performs the required task is called functional design.

A Field Programmable Gate Array (FPGA) is electronic device. A device built as an array of configurable logic elements (LE). Each logic elements can be configured to work in combination or sequential functions. Modern FPGAs integrate other useful things features such as built-in multipliers and high-speed (I/O), data converters including analog to digital converters, large RAM arrays (random access memory), and processor. All these features allow you to create complex system-on-Chip (SoC) hardware that provides options such as create a specific custom Central Processing Unit (CPU), the purpose of which is executing multiple statements. An existing printed circuit board (PCB) such as B. DE10 System on Chip (SoC) and allows you to build and craft using the Hard Processor System (HPS) implements two embedded systems in real time, each working independently or together with its own processor (CPU).

MIMAS V2 is a feature-rich and cost-effective FPGA development board powered by Xilinx Spartan-6 FPGA. MIMAS V2 is specifically designed for experimenting and learning system design using FPGAs. This development board features a SPARTAN XC6SLX9 CSG324 FPGA with 512MB DDR SDRAM. A USB 2.0 interface allows quick and easy configuration downloads to the integrated SPI flash. You don't need to buy expensive programmers or special downloader cables to download bitstreams to your board. Fig1. displays the FPGA development board

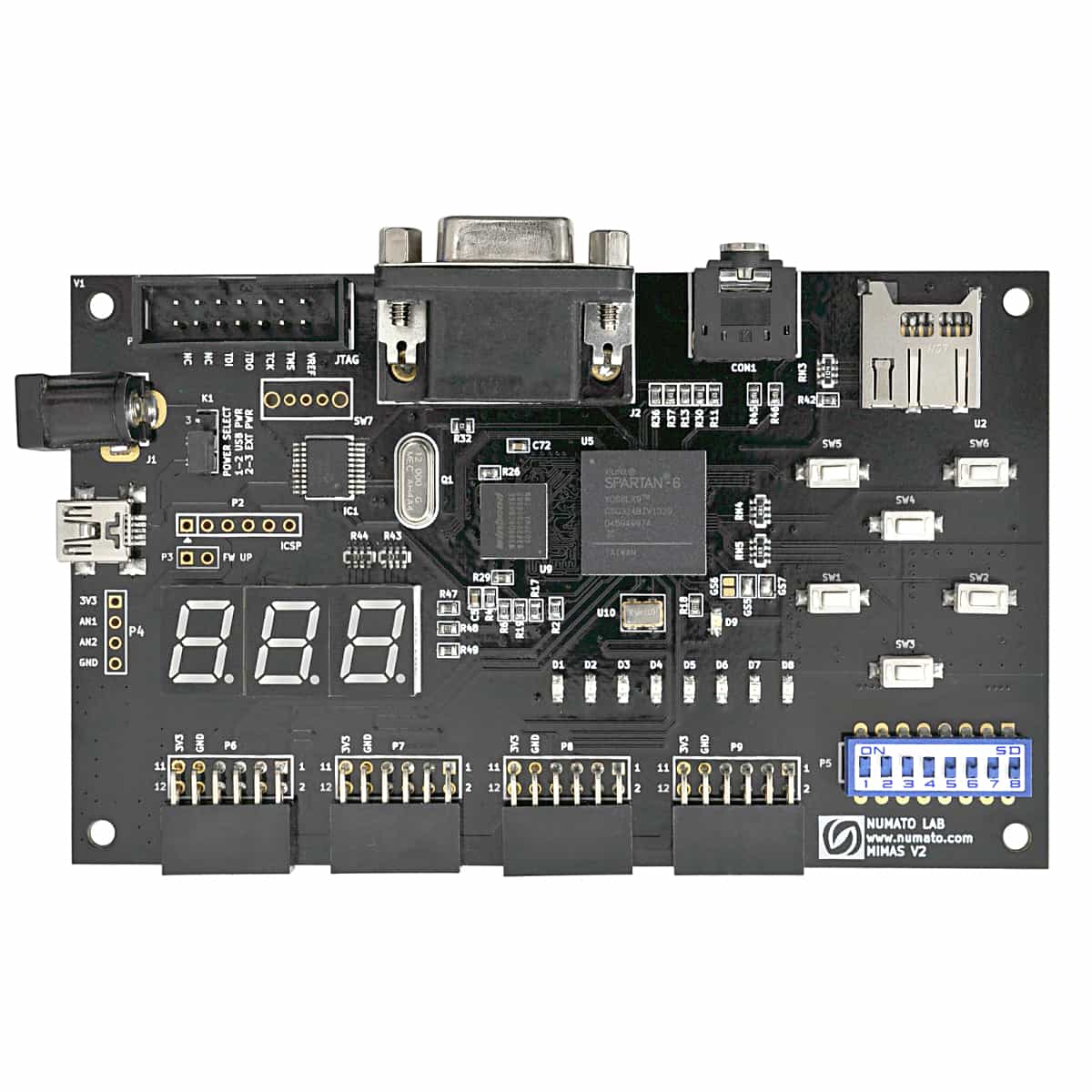


Fig. 1 FPGA Development Board [17]

*A. Features of FPGA*

FPGA: Spartan XC6SLX9 in CSG324 package.

DDR Memory: 166MHz 512Mb LPDDR (MT46H32M16LF/W949D6CBHX6E).

Flash memory: 16 Mb SPI flash memory (M25P16).

USB 2.0 interface for On-board flash programming.

FPGA configuration via JTAG and USB 8 LEDs, six push buttons and 8-way DIP switch for user-defined purposes.

VGA connector, Stereo Jack.

Micro SD card adapter.

Three-digit seven segment displays.

32 IOs for user-defined purposes.

Four 6×2 expansion connectors.

Onboard voltage regulators for single power rail operation.

V. SPI COMMUNICATION PROTOCOL

SPI is a common communication protocol used by various devices. For example, the SD card reader module, RFID card reader module, and 2.4 GHz wireless transceiver all use SPI to communicate with the microcontroller.

A unique advantage of SPI is the ability to transmit data without interruption. Any number of bits can be sent and received in a continuous stream. In I2C and UART, data is sent in packets limited to a certain number of bits. Data is corrupted during transmission because start and stop conditions define the beginning and end of each packet.

Devices that communicate via SPI have a master-slave relationship. The master is the controlling device, and the slaves (usually sensors, displays, or memory chips) receive instructions from the master. The simplest configuration of SPI is a single master single slave system, but one master can control multiple slaves. Fig.2. describes about the connection between master and slave.

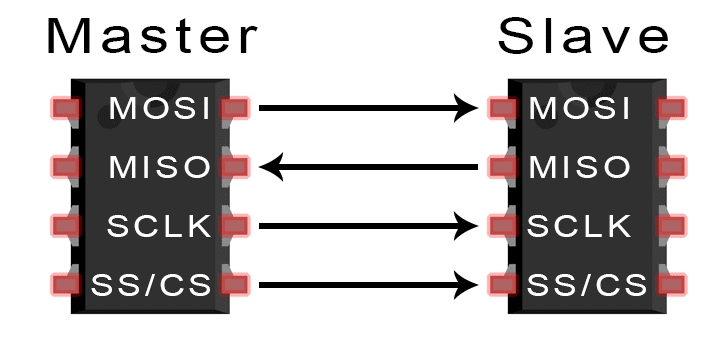


Fig. 2 MOSI [16]

*A. Pin Functions of SPI*

MOSI (Master Output/Slave Input) – master send data to slave.

MISO (Master Input/ Slave Output) – slave send data to master,

SCLK (Clock) – clock signal line.

SS/CS (Slave Select/Chip Select) – master select to which slave to send data to.

*B. MISO AND MISO*

The master sends bitwise serial data to the slave over the MOSI line. The slave receives data sent by the master on the MOSI pin. Data sent from a master to a slave is normally sent most significant bit first.

The slave can also send data serially back to the master over the MISO line. Data sent back from the slave to the master is usually sent least significant bit first.

*C. Steps*

The clock signal gets output by the master. The master drives her SS/CS pins to a low voltage state, enabling the slave. The master sends data bit by bit to the slave over the MOSI line. The slave reads the received bits. When a response is required, the slave sends the data back to the master bit by bit over her MISO line. The master reads the received bits.

*D. Advantages*

There are no start and stop bits, so data can be streamed continuously without interruption. No complex slave addressing system like I2C. Higher data transfer rate than I2C. Separate the MISO and MOSI lines so that data can be sent and received simultaneously.

VI. PROPOSED WORK

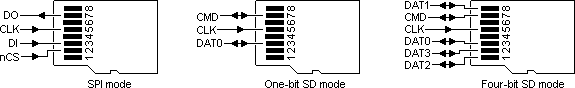


Fig.3. SPI mode pin diagram

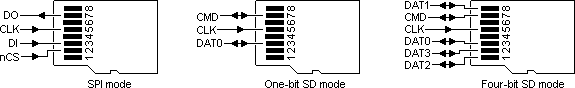


Fig.4. 1-bit SD mode pin diagram

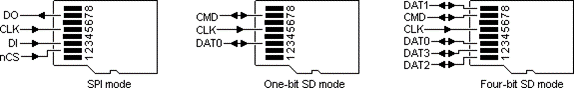


Fig.5. 4-bit SD mode pin diagram

Table III. Comparison of 1-bit SD mode, 4-bit SD mode, SPI mode

|  |  |  |  |
| --- | --- | --- | --- |
|  | **1-bit SD mode** | **4-bit SD mode** | **SPI mode** |
| **Input Signal** | 1 CLK PIN,  1 DATA PIN | 1 CLK PIN,  1 CMD PIN,  4 DATA PINS | MOSI,  MISO, CS, CLK |
| **Minimum Frequency** | 0 MHz | 0 MHz | 1 kHz |
| **Maximum Frequency** | 25 MHz | 25 MHz | 75 MHz |
| **Bit Rate** | 25 Mbps | 100 Mbps | 25 Mbps |

While comparing these 3 modes (1-bit SD mode, 4-bit SD mode, SPI mode) 4-bit SD mode is faster than 1-bit SD mode and SPI mode. The bit rate for 4-bit SD mode is faster and efficient for executing commands like read, write operations.

*A. FPGA Controller Structure*

The FPGA block diagram for proposed 4-bit SD mode is shown in Fig. 6. In 4-bit SD mode , the 1 clock, 4 Data pins.

The Clock Control module is necessary for data transmission.

The Command Control module ensures all the functions of 4-bit SD mode. SD card has fixed memory location of 512 bytes, so 2 FIFO buffers are added for storing temporary data that will transmit and receive respectively.

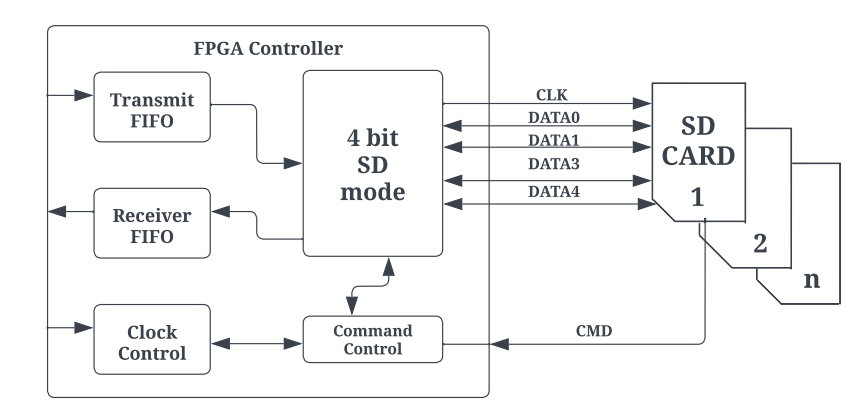


Fig. 6 FPGA block diagram for 4-bit SD mode

*B. SD Card Initialization*

There are special steps to initialize an SD card in SPI mode. All SD card slots have a switch on the back to indicate when a card is inserted. You need to pull up the card select pin. This will deselect the card. At least 76 to 160 pulses must be sent to the clock for SD to initialize itself. SD does not have an internal clock source.

Source code: Sdcard\_controller.v

Command 0 is a software reset that hibernates the SD card. Once in this state, it can be set up to run in SPI mode. Only one NCR is required.

cmd\_out <= 56`hFF\_40\_00\_00\_00\_00\_95

Command 8 is to check if you are using the correct card. Otherwise, this particular program will always return to the beginning. This part of the initialization procedure is mandatory.

cmd\_out <= 56'hFF\_48\_00\_00\_01\_AA\_87;

After the R1 response, we get the R3 response. All you need to know is that the last byte you receive must be (hex) AA. This indicates that an SD card version 2 (SDHC) was detected. Only one NCR is required. Finally, the ACMD41 command puts the SD card into SPI mode. Only one NCR is required.

cmd\_out <= 56'hFF\_69\_40\_00\_00\_00\_01;

The idle flag remains set during the first run. If the flag is cleared, the initialization process is complete. Otherwise, command 55 is issued. The quirk of the ACMD command is that all commands are followed by command 55.

cmd\_out <= 56'hFF\_77\_00\_00\_00\_00\_65;

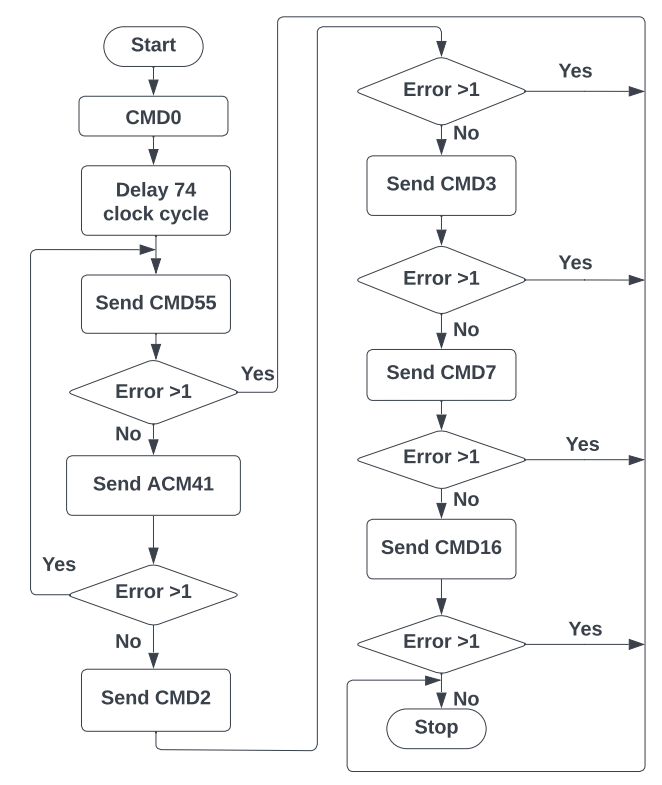
Note: CRC, NCR, or null arguments are normally sent as 0xFF. The reason the data stays high is that the SD card works that way. When the SD card is busy its data output pin goes low and when it is ready the data output pin goes high. This is very useful when writing to cards. There is no need to send commands to read the status register. **

Fig. 7 Flow Chart of SD card initialization

*C. SD Card Write Data*

The data can be written to the ‘Memory Core’ of the SD card using the commands given below followed by actual data;

WRITE\_BLOCK – Write data to a single block (512 bytes).

**WRITE\_BLOCK**

In the SD card a block is always considered as consecutive 512 bytes memory locations. If a block starts from the 2000th memory location it needs to be written with some data using the WRITE\_BLOCK command. The command packet should be like as shown below:

1st byte (command) – 0x18.

2nd to 5th byte (argument) – 0x000007d0 (Even if there are no arguments for other commands, this field should be set to zero).

6th byte (CRC) – any value.

7th byte NCR.

Once the command has been sent to the FPGA should receive the R1 response. All the bits are supposed to be zero. After receiving the zero valued R1 response byte, the FPGA can send the data to be written into SD card. The length of data should be 512 bytes even though the actual data has a smaller number of bytes.

The 512-byte data is preceded by a Data Token byte. Next, it should be terminated with 16-bit CRC byte. This 1+512+2=515 bytes will form a data packet. The data token are all the bits except the LSB which is set to 1(0xFE).

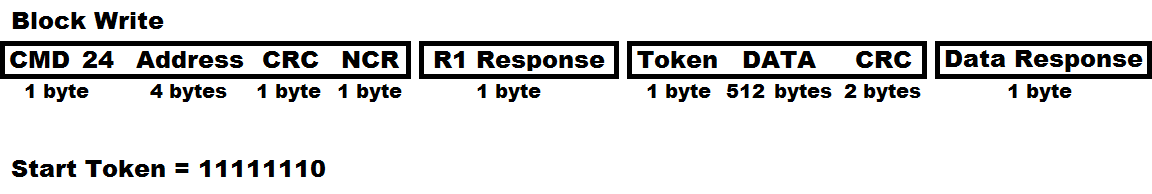


Fig. 8 Data packet format [9]

From Fig.8. the command to issue would be 24, dummy bytes are then sent until a clear R1 response is received, a dummy byte and then token byte (11111110b) is sent, the 512 bytes of data are sent followed by two CRC’s, finally a data response is received to say if the data was successful, dummy data is continuously sent until the correct response is received indicating the write has completed, this would be receiving any data apart from zero. No erase command for the SD card, it is done by itself in hardware.

cmd\_out <= {16'FF\_58, address, 8’FF};

After the data response is received the SD card is ready to write, to do this SD card has to be de-asserted, clocked eight times (1 byte) and then reasserted before the status can be checked. The above section of program may sometimes work and it is often how datasheets are to be interpreted, however for the correct procedure the SD card has to be deselected to initiate the write sequence.

When the SD card is busy it will pull the data output low (only when it has been selected), when it pulls high it means that it is ready. The program below is an alternative method to that above, while they both achieve the same thing, we are ideally looking for 0xFF, this is the correct procedure.

For writing the next data block the WRITE\_BLOCK command should be sent again.

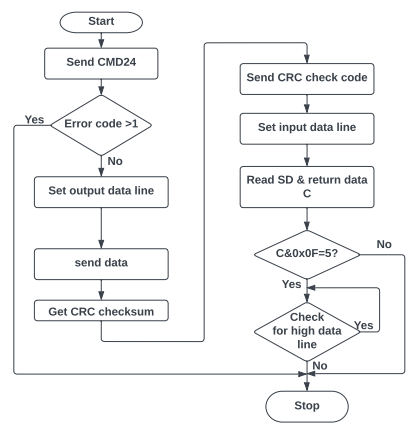


Fig. 9 Flow chart of SD card write data

*D. SD Card Read Data*

The data can be read from the ‘Memory Core’ of the SD card using the commands given below;

READ\_BLOCK – Read a single block (512 bytes) from SD card.

***READ\_SINGLE\_BLOCK***

In the SD card a block is always considered as consecutive 512 bytes memory locations. If a block starting from the 2000th memory location needs to be read using the READ\_SINGLE\_BLOCK command. The command packet should be like as shown below;

1st byte (command) – 0x51.

2nd to 5th byte (argument) – 0x000007d0 (Even if there are no arguments for other commands, this field should be set to zero).

6th byte (CRC) – any value.

7th byte NCR.

Once the command has been sent, FPGA should receive the R1 response. All the bits are supposed to be zero. After receiving the zero valued R1 response byte, the FPGA can read the data from the SD card. The data of 512 bytes is sent by SD card in response to each READ\_SINGLE\_BLOCK command. cmd\_out<= {16h’FF\_51, address,8’hFF};

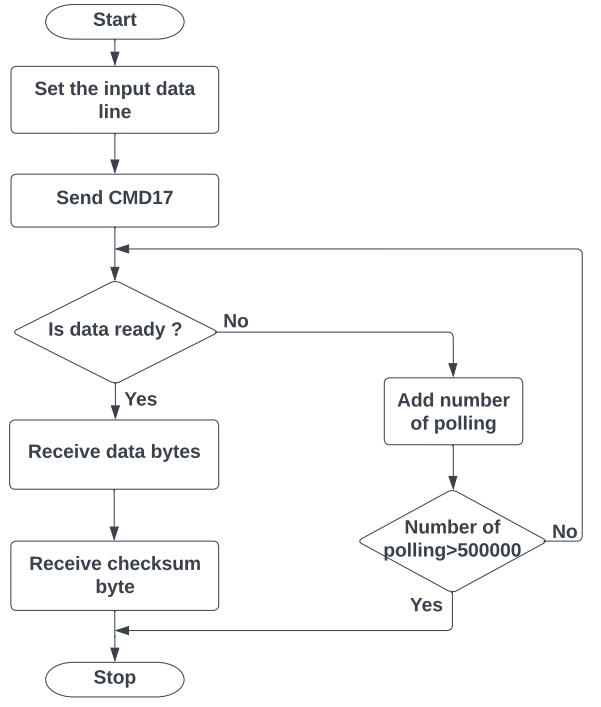
**

Fig. 10 Flow chart of SD card read data

*E. SD Card Controller Working*

**Input**

Clock - 25Mhz

Reset - Active high reset

Din [7:0] - Data in [write operation]

Address [31:0]- Sector address [Read/Write operation]

Wr - Write enable

Rd - Read enable

Multi\_sector\_en- Multiple sectors read Henable

I\_blk\_num - Read total number of blocks in multi sector mode

Miso - Read total number

**Output**

Cs - chip select

Mosi - FPGA send command/data to be shared

Sclk - spi clock

byte\_counter- byte count till 512 bytes

Dout, recv\_data- data out from Sd card

status - state changes

byte\_available- ensure the valid data by data enable

Reading - data is reading from Sd card

Ready - ready to send the read/write command

Read\_for\_next\_byte - each byte has 8bits. Ready to send next byte

Read\_done - enable every 512 bytes completed

*F. FAT32 File System Format*

The FAT32 file system is stored or written to memory cores. There are certain defined sectors at the beginning which are then followed by Clusters which is shown in Fig. 11.

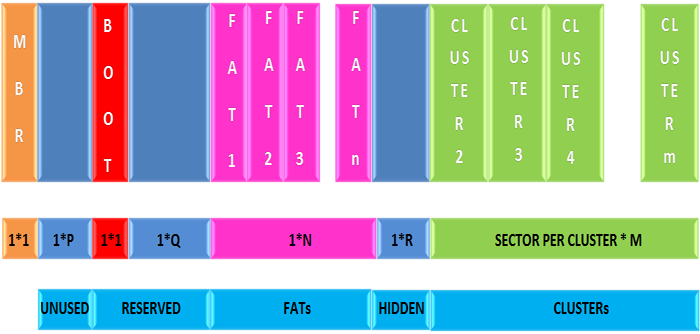


Fig. 11 Format of FATR32 file system in memory card [9]

MBR also known as Master Boot Record is the first sector. This follows significant number of unused sectors. These sectors are followed by reserved sectors among which the first sector is the BOOT sector where reserved sectors are followed by the FAT sectors. The number of FAT sectors depends upon the size of the file system. The FAT sectors are followed by few hidden sectors which is followed by the Clusters.

*G. Logic for reading a file from FAT32 file system*

A file can be read from the FAT32 formatted file system. It can be found that every process finally ends with a sector read. This sector read from the memory core of the SD card can be achieved by using the READ\_SINGLE\_BLOCK command from the SD Command Layer alone.

sdcard\_fat32\_read. V

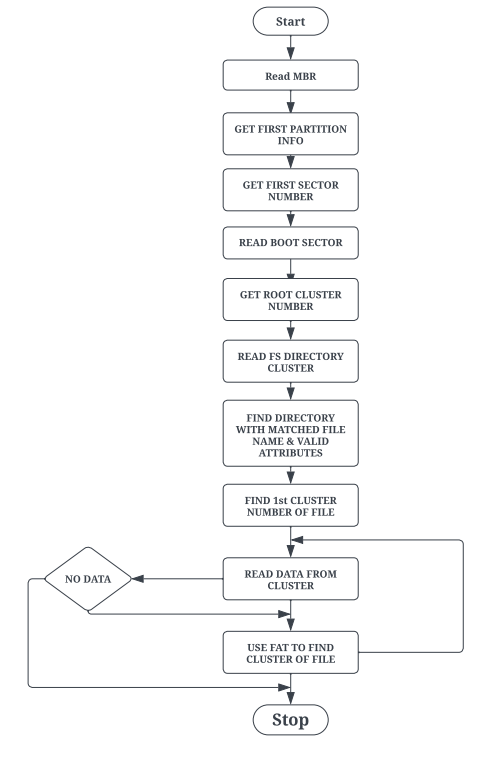


Fig. 12 Algorithm for Reading file from FAT32 file system [9]

FAT32 are sectors in which each 32 bits holds the cluster number of clusters. Each 32 bits point towards a particular cluster. A cluster normally has 512 bytes, so there will be 128 cluster pointers inside the sector. This forms the File Allocation Table 32 FAT32.

The number of the next cluster pointer inside the FAT32 can be calculated by using the following equation.

Number of FAT sectors in the next cluster pointer = number of first sectors in partition + number of reserved sectors + ((current number of clusters \* 4)/bytes per sector).

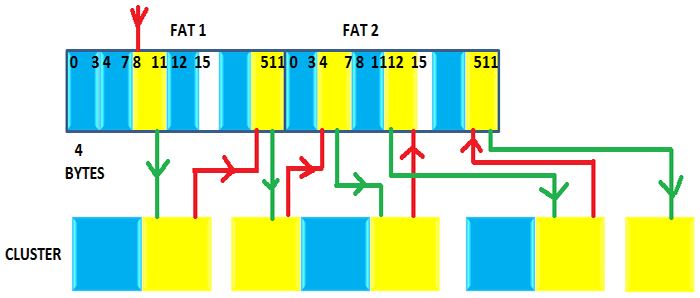


Fig. 13 Read Scrambler file across flash ‘Memory Core’ [9]

From Fig. 13 the yellow indicates the clusters that contain the data for each file in FAT32 and the corresponding cluster pointers. The red line indicates searching for the next cluster pointer that matches the current cluster, and the green line indicates searching for the next cluster using the cluster number stored in the FAT32 cluster pointer.

VII. RESULTS

Table IV. Comparison between existing methods and our method

|  |  |  |  |
| --- | --- | --- | --- |
| **Proposed Work** | **LUT (Look-Up Table)** | **Flip Flop** | **Clock period** |
| Spartan 6 | 916 | 464 | 5.35 ns |
| Artix-7 | 653 | 413 | 3.254 ns |
| Existing Method |  |  |  |
| [1] Artix-7 | 414 | 283 | - |
| [3] Altera’s Cyclone II | - | - | 4.15 ns |
| [4] Altera Stratix IV | - | - | 0.005 s |
| [14] Cyclone V | - | - | 27360 ns |

Spartan 6 FPGA has 530 Slice Registers, 916 LUT’s, 464 Flip Flops, 5.35 ns clock period, 186.925 MHz frequency where at same Artix-7 has 519 Slice Registers, 653 LUT’s, 413 Flip Flops, 3.254 ns clock period, 307.342 frequency when compared to other existing methods our proposed results are better resource utilization.

VIII. CONCLUSION

In this paper, we proposed an FPGA controller. Allows SD card writing and reading with SPI protocol. SPI transmit mode is required. Easily expandable storage system add number SD cards contain sharing the same clock and data signals different chip select signal for each card was suggested the architecture involves accessing each card in sequence, but it is allowed. Availability of large amounts of memory .

IX. REFERENCE PAPERS

[1] Dumitrel Cătălin Costache, Lucian Andrei Perișoară, Adriana Florescu, *FPGA Implementation of SD Card Controller Using SPI Communication,* Conference Paper, June 2020

[2] 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.

[3] Omar Elkeelany, Vivekanand S. Todakar Data Archival to SD Card Via Hardware Description Language,” IEEE Embedded Systems Letters, Vol. 3, No. 4, pp. 105-108, Dec. 2011.

[4] Gul Munir Ujjan, Abdul Malik, Mohd Zaid Abdullah, Shakil Ahmed, “Implementation of 4-bit data transmission for accessing SD card with FPGA Embedded Soft Processor;” ICIIT '19, February 20–23, 2019, Da Nang, Viet Nam

[5] Jiayi Qiang, Yong Gu and Guochu Chen, “FPGA Implementation of SPI Bus Communication Based on State Machine Method,” Journal of Physics: Conference Series 1449 012027, 2020.

[6] Pallavi Polsani, V. Priyanka.B, Y. Padma Sai, “Design & Verification of Serial Peripheral Interface (SPI) Protocol,” IJRTE, Volume-8 Issue-6, March 2020.

[7] Fareha Naqvi, “Design and Implementation of Serial Peripheral Interface Protocol Using Verilog HDL,” International Journal of Engineering Development and Research 2015.

[8] Aviraj Ghanekar, Braj Kishor, Sachin Bandewar, “Design and Implementation of SPI Bus Protocol,” International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineerinf Vol. 5, Issue 5, May 2016.

[9] [Interfacing SD Card with AVR Microcontroller- (Part 38/46) (engineersgarage.com)](https://www.engineersgarage.com/interfacing-sd-card-with-avr-microcontroller-part-38-46/).

[10] Xilinx Inc., “AC701 Evaluation Board for the Artix-7 FPGA User Guide UG952 (v1.4)”, 6 Aug. 2019.

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

[12] SD card association. 2006. SD specification part-1 physical layer simplified specification version 2.00.

[13] SanDisk Corporation. 2003. SanDisk secure SD card. Product Manual. Version 1.9 document no 80-13-00169.

[14] Víctor Asanza, Alisson Constantine, Stephany Valarezo, and Enrique Peláez, “Implementation of a Classification System of EEG Signals Based on FPGA,”

[15] SD Card Association, “SD Part 1, Physical Layer Simplified Specification, Version 7.10”, 25 March 2020.

[16] [Basics of the SPI Communication Protocol (circuitbasics.com)](https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/).

[17] [Mimas V2 Spartan 6 FPGA Development Board | Numato Lab](https://numato.com/product/mimas-v2-spartan-6-fpga-development-board-with-ddr-sdram/).