**SD CARD INTERFACE USING FPGA FOR MULTIMEDIA APPLICATIONS**

Sakthekanna M S1, Udaya Krishnan M2, Yogavignes B M3, \*Dr . Srigitha S Nath4

1,2,3,4Department of Electronics and Communication Engineering, Saveetha Engineering College,Chennai

[1srivisakthi@gmail.com](mailto:1srivisakthi@gmail.com)

[2udayakrishnan.ece.sec@gmail.com](mailto:2udayakrishnan.ece.sec@gmail.com)

[3yogavignesbm@gmail.com](mailto:3yogavignesbm@gmail.com)

[4,\*hod.ece@saveetha.ac.in](mailto:4navaneethans@saveetha.ac.in)

***Abstract—*** **FPGA (Field Programming Gate Array) is based on chips, this application is used to process audio, video and image. At same time, applications require more memory to deal with additional data. Onboard memory can be used to meet the needs of FPGA systems but cannot be extended by adding just a few cards. More practical are SD cards (Secure Digitals) which are microcontrollers that can read and write to SD cards. The main objective is to develop a storage solution for FPGA that is low-cost, non-volatile, flash memory, removable, portable and simple to use for storing big size of files. The hardware is designed in Verilog HDL and implemented in FPGA. Verilog is used for all data access from the SD card, avoiding the requirement for an on-chip microcontroller or general-purpose.** **Spartan 6 (XC6SLX9-3csg324) FPGA is utilized. The FPGA is powered by a 5V source and has a built-in oscillator frequency of 100 MHz . In this project, a 4GB micro SDHC (class 6) card from Strontium is utilized. The SD card formats FAT32. This project's ultimate goal is to read a BMP picture file from an SD card.** **Before connecting, the SD card formats FAT32. The FAT32 code is built to connect with the SD card.**

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

I. INTRODUCTION

Secure Digital (SD) cards are non-volatile proprietary cards. A new generation semiconductor-based memory card is a portable or stationary electronic device designed primarily for data storage. SD cards offer the advantages of being tiny, having big storage capacities and quick data bits, being inexpensive, having high energy efficiency, tremendous flexibility, and outstanding safety. SD cards along with USB memory sticks, are portable storage devices that are most commonly used in digital systems such as microprocessors, microcontrollers, and Digital Signal Processor (DSP) or Field Programmable Gate Array (FPGA) chips, digital cameras, laptops, mobile phones, and embedded systems [1].

The development of an SD Card Association is the solution to user needs and requirements (SDA). Its purpose is to promote and develop the SD card standard, which includes the card type, speed class, electrical interface, and communication protocol. There are three SD card communication protocols supported: 1-bit SD mode, 4-bit SD mode, and Serial Peripheral Interface (SPI) mode [1].

SD cards are flash memory chips that are based on the latest generation of semiconductor storage devices. SD card with a large capacity, good file transfer speeds, high flexibility, and excellent mobile security [2].

II. LITERATURE REVIEW

Dumitrel Catalin Costache et al (2020) that the SPI protocol is used by an FPGA controller to write to and read from an SD card. SPI transfer mode is required for simple storage system expansion. When the number of SD cards is increased, each card must share the same clock and data signals as well as different chip select signals. This requires accessing each card one at a time, yet it allows for huge amounts of memory.

Gul Munir Ujjan et al (2019) presented that a for managing SD cards in 4-bit SD mode, a primary hardware architecture based on FPGA and integrated NIOS-II processor is created. 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. For comparison, these commands are accessible in SD 1-bit mode. SD 4-bit read performance is around 67% better than SD 1-bit read throughout overall. Thus, SD 4-bit mode is 80% quicker than SD 1-bit mode in write operations. Implementing a fast technique that performs multi-block writes and reads and computes 16-bit CRC write instructions can greatly increase throughout. This needs more hardware resources and consequently more expenditures. The suggested firmware will be useful for systems that deal with enormous volumes of data, such as real-time video capturing.

Pallavi Polsani et al (2020) presented that similar to external analog to digital converters, digital-to-analog converters, and EEPROMs, serial synchronous communication through the use of communication between master and slave devices is also utilized for producing connectivity between microcontrollers and various devices included. There are basically two different types of protocols: 1) SPI and 2) Inter-I2C. Both protocols are ideally intended for communication between integrated circuits for onboard neighborhood communication. SPI has become the most widely used protocol for both low and medium-speed communication within and between chips transfer of data streams. SPI interface protocol with one master and single slave 8-bit data transfer and complete configuration. System Verilog was used to verify and implement the SPI design. Shows that the coverage code and functionality are correct. For synthesis, the whole RTL is built in Verilog, while the verification architecture is defined in System Verilog and the implementation is done on Spartan 3E.

Jiayi Qiang et al (2020) presented that the SPI bus is a type of synchronous, full-duplex, serial interface data bus line with a simple protocol and a high transmission speed. Based on the properties to allow device development and experimentation at a quick speed, parallel enhanced computation with FPGAs is employed. This describes the construction and operation of SPI. The communication bus examines its sequence and four modes of operation and makes use of this information. Module circuit is a machine mechanism that delivers the SPI bus interaction feature on an FPGA. The Verilog hardware description language is utilized to create the SPI, 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 in a customizable FPGA-based data extractor device, a bidirectional hardware-based component for SD card design is built. In all, 5000 blocks may be written in a single second and read from an SD card in 1.051 seconds. The previous attempt took 60 seconds for the same block size. This distinction is due mostly to the usage of software-based methods and specialized finite-state machine architectures seen in related research. It demonstrates that actual data speeds of 25 Mb/s are possible. This letter describes a real-time architecture for data archiving on SD cards for use in distant site data concentrator operations such as SG. In the future, scaling concerns such as the effect of larger SD cards and SD cards from multiple manufacturers will be considered.

Zinlin et al (2010) presented that it allows for the rapid production of relevant files on the SD card. Then, for SD memory card reader wants, it creates an FPGA-primarily based completely SD card reader system structure with different specific parts for design and improvement. The equipment was put through practical tests. According to the test findings, the device can inspect and write data to SD cards, which is sufficient to meet the FPGA device requirements for outside garage devices. At the same time, this concerned the SD card reader generation, which might be carried out to a number of FPGA-based entirely device. As a result, the apps used in this period must also be more capable.

III. SD PRINCIPLE

*A. Mode of Transmission*

Three transmission modes are supported by SD cards: SPI (different parallel input and output), 1-bit SD (different command and data channels, one-time transfer type), and 4-bit SD modes (with extra pins and some reset pins 4-wide parallel transmission) are supported. SD card mode access is quicker and more efficient. The transmission rate of 4-bit SD data transfer mode is 0-100Mbps, while data transfer from 1-bit SD mode is up to 0-25Mbps. SD transfer mode using 4-bits has a higher transmission rate, but the system's complicated structure and timing make it challenging to create. [1].

*B. Bus Protocol of SD*

The SD communication idea is easy, master-slave mode. This configuration frequently contains a master and numerous slave controller with 3-6 lines. CMD (control line), CLK (clock line), and DAT0-3 (data line) are the ones.

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. System File of SD*

The FAT16 data file on SD card is divided into four sections. PBR (Partition Boot Record), FAT (File Allocation Table), FDT (File Directory Table), and Data Section. A partitioned boot file normally has four factors: BPB (BIOS Parameter Record Block), hard drive flag record book, partition boot record code region, and end Flag 55AA. PBR includes a BIOS parameter setting block. It should be noted that certain characteristics are more essential than others. These values can be used to calculate the address of the FAT, file path column, and data region (which region in the SD map).

*D. Command Format for SD*

All SD internal memory commands are six bytes long, which includes a 1-byte command code, 4-byte command inputs, and checksum bits. Even when no parameters are specified, the host must provide 4 bytes of arguments when transferring instructions towards the SD card. In this situation, the argument can be of any value and will be disregarded by SD card.

Table II SD card commands for SPI mode

**Commands Functions**

CMD 0 Software been reset

CMD 1 Initialization starts

CMD 3 Request card send back the RCA address

CMD 7 Card entering the state

CMD 9 Read CSD register

CMD 10 Read CID register

CMD 12 Stop read data

CMD 16 Change of size in read/write

CMD 17 Read command for single command

CMD 18 Read command for multiple command

CMD 23 Amount of block sent

CMD 24 Write for single block

CMD 25 Write for multiple block

CMD 32 Erase start block

CMD 38 Erase command

CMD 55 ACMD <n> leading command

CMD 58 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 FPGA (Field Programmable Gate Array) is electronic device. A device constructed of an array of programmable logic components (LE). Every logic element can be programmed to perform sequential or combination tasks. Other essential characteristics of modern FPGAs include built-in multipliers and high-speed (I/O), data is being processed such as analog to digital converters, big RAM arrays (random access memory), and processing units. All of these characteristics enable you to design complicated system-on-chip (SoC) equipment with possibilities like creating a specialized customized Central Processing Unit (CPU) for the task of running numerous statements. A pre-existing printed circuit board (PCB) such as B. DE10 System on Chip (SoC) and enables you to design and construct using the Hard Processor System (HPS) utilizes two embedded systems simultaneously in real time, each functioning independently or collaboratively with their own processor (CPU).

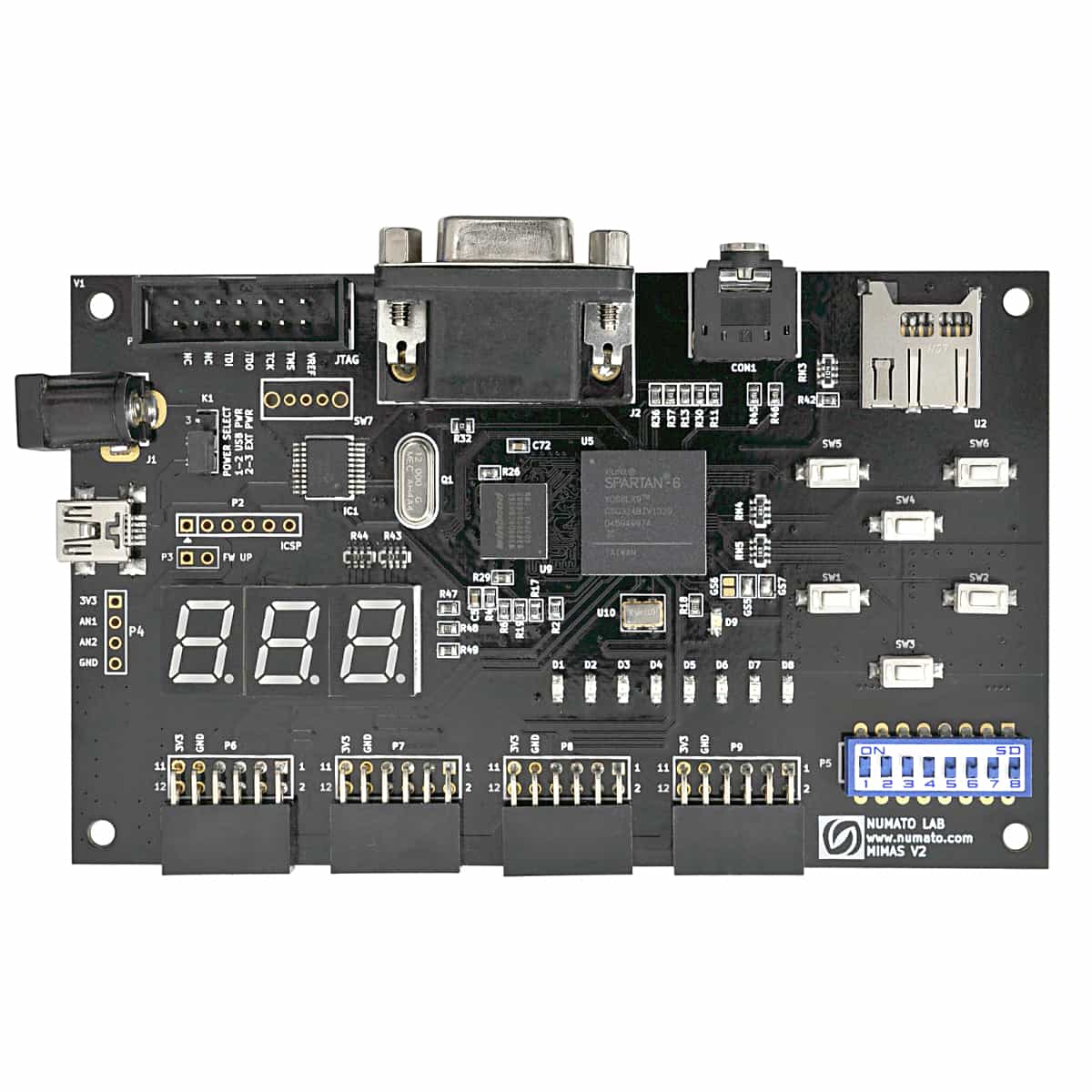
MIMAS V2 is just a functionality and low-cost FPGA development board powered by a Xilinx Spartan-6 FPGA. MIMAS V2 is particularly built for testing with and understanding system design via FPGAs. A SPARTAN XC6SLX9 CSG324 FPGA with 512MB DDR SDRAM powers this development board. A USB 2.0 interface enables fast and simple configuration uploads to the inbuilt SPI flash. To download bitstreams to your board, you do not need to purchase costly programmers or special downloader connections. Figure 1 depicts the FPGA development board. 

Fig. 1 FPGA Development Board [17]

*A. Features of FPGA*

Spartan XC6SLX9 FPGA (CSG324 configuration).

DDR memory (166MHz, 512Mb, LPDDR) (MT46H32M16LF/W949D6CBHX6E). SPI flash memory with a capacity of 16 Mb (M25P16).

USB 2.0 port for programming on-board memory.

FPGA setup via JTAG as well as USB 8 LEDs, 6 push buttons, as well as an 8-way DIP switches for custom uses.

Stereo jack, VGA connector.

Adapter for Micro SD cards.

Seven segment displays with three digits.

32 IOs per user-defined functions.

There are four 62 expansion connections.

Power supplies onboard for single power rail usage.

V. COMMUNICATION PROTOCOL OF SPI

SPI is just a standard communication protocol that is utilized by many devices. An SD card reader module, RFID card reader module, and wireless 2.4 GHz transceiver, for example, all connect with the microcontroller using SPI.

The capacity to send data without interruption is a distinct characteristic of SPI. In a constant stream, any number of bits can be delivered and collected. Data is transferred in packets of a particular amount of bits in I2C and UART. Because start and stop criteria specify the beginning and conclusion of each packet, data gets distorted while transmission.

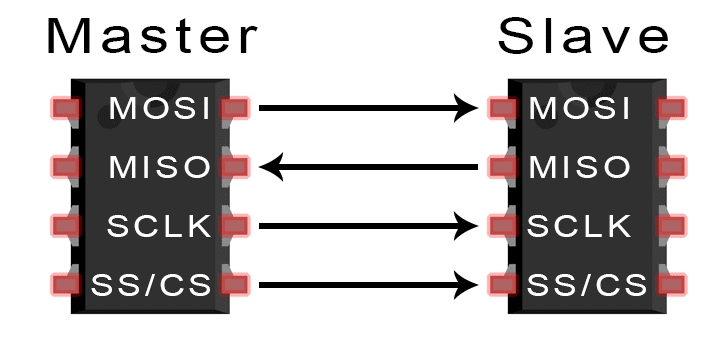
SPI-enabled devices have a master-slave relation. The controlling device is the master, and the slaves (often detectors, screens, or flash memory) receive commands from the master. The most basic SPI configuration is a single master, single slave network, however one master can manage several slaves. Figure 2 depicts the relationship between master and slave. 

Fig. 2 MOSI [16]

*A. Pin Functions of SPI*

Master Output/Slave Input (MOSI) - the master transmits data to the slave.

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

Clock (SCLK) - clock signal line.

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

*B. MISO and MOSI*

Over the MOSI line, the master sends bit - wise data bits to the slave. On the MOSI pin, the slave gets data transmitted by the master. Data transmitted from the a master to the a slave is usually sent using the most significant bit first.

The slave also can sequentially transmit information back towards the master via the MISO line. Data returned as from slave towards the master is typically delivered with the smallest bit first.

*C. Steps*

The master generates the clock frequency. The slave is activated when the master pushes the SS/CS pins to a reduced voltage condition. Over the MOSI line, the master transmits information towards the slave bit by bit. The received bits were examined by the slave. Whenever a response is needed, the slave delivers the data bit-by-bit back to the master over her MISO line. The received bits are analyzed by the master.

*D. Advantages*

Since there are no beginning and end bits, data can really be delivered continuously. There is not a complicated slave specifically addressing mechanism as I2C. Data transmission rate is greater than I2C. Divide the MISO and MOSI lines such that data can be transmitted and recieved on the same time.

VI. PROPOSED WORK

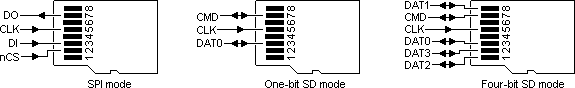


Fig. 3 SPI mode pin diagram

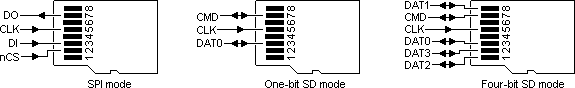


Fig. 4 one-bit SD mode pin diagram

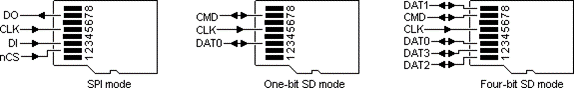


Fig. 5 four-bit SD mode pin diagram

Table III. Comparison of one-bit SD mode, four-bit SD mode, SPI mode

|  |  |  |  |
| --- | --- | --- | --- |
|  | **One - bit SD mode** | **Four - 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 (one - bit SD mode, four - bit SD mode, SPI mode) four - bit SD mode is faster than one - bit SD mode and SPI mode. The bit rate for four - bit SD mode is faster and efficient for executing commands like read, write operations.

*A. Control Structure of FPGA*

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

The Clock Control module is main goal is to transfer data.

The Command Control package practically 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.

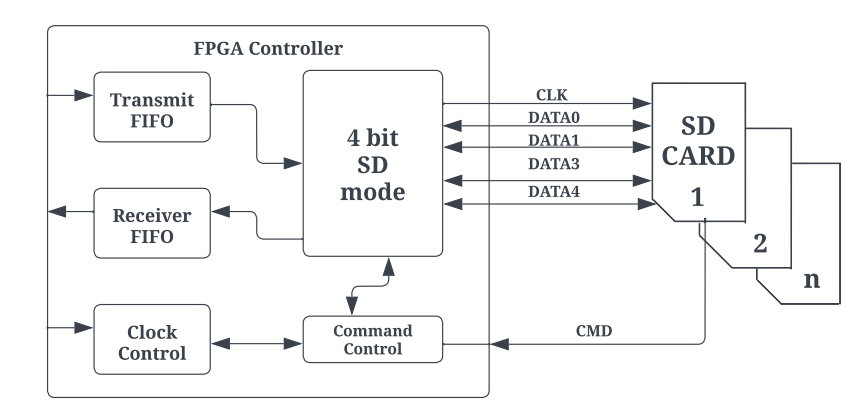


Fig. 6 FPGA block diagram for 4-bit SD mode

*B. SD Card Initialization*

To start an SD card increased actions are required. A switch on the rear of every SD card slot indicates whenever a card is inserted. You must enter the card choose pin. It will disable the card. SD must always be initialized with at minimum 76 - 160 pulses transmitted to the clock. SD lacks an internal clock primary source.

Source code: Sdcard\_controller.v

Command 0 is just a computer reset which puts the SD card to rest. When it reaches this condition, it may be configured to operate in SPI mode. There is only one NCR needed.

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;

Then to get R3 reaction after the R1 reaction. The one and only thing you need to understand is that the final byte you get has to be (hex) AA. This signals the detection of an SD card version 2 (SDHC). There is only one NCR necessary. Furthermore, the ACMD41 instruction configures the SD card to operate in SPI mode. There is only one NCR necessary.

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

During the initial run, the idle flag remains set. This activation procedure is complete if the flag is cleared. Alternatively, command 55 is delivered. The ACMD command has a feature in that all commands are preceded with command 55..

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

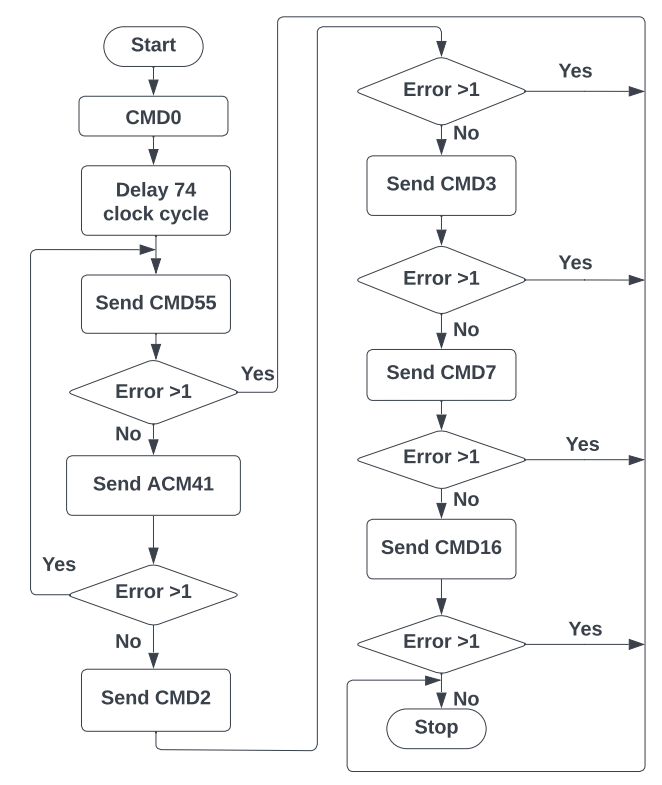
CRC, NCR, and null arguments are often transmitted as 0xFF. The SD card operates in this manner, which explains why the data remains high. Whenever the SD card gets busy, the output data pin goes to ground, and when it is ready, it goes too high. This is helpful when writing on cards. To read a state register, no instructions are required. **

Fig. 7 SD card initialization flow chart

*C. Write Data for SD card*

The data could well be transferred to the SD card's 'Memory Core' using the instructions shown below, followed by real data.

WRITE\_BLOCK–Single block of data is written(512 bytes).

**WRITE\_BLOCK**

A block on an SD card has always been viewed as a collection of 512-byte memory addresses. When a block begins at memory address 2000, it must be written with data using the WRITE BLOCK command. The command package must appear like this:

0x18 is the first byte (command).

0x000007d0 - 2nd up to 5th byte (argument) (This field must be set to 0. even though there are no parameters for those other commands.

(CRC) is the 6th byte with any random values.

7th byte NCR.

As once command has really been transmitted to the FPGA, the R1 responds should be received. Each of the bits should be zero. The FPGA can deliver the data to also be written onto the SD card after obtaining the zero-valued R1 response byte. Even if the actual data contains less bytes, the data length should be 512 bytes.

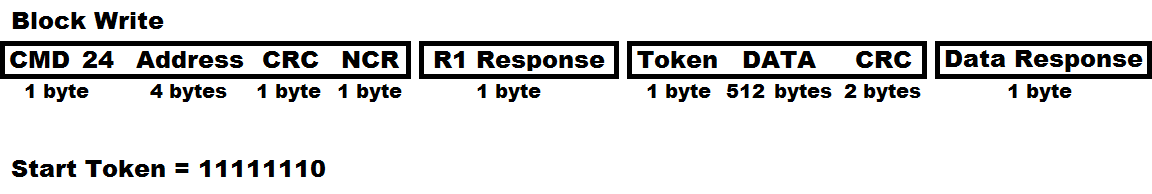
The Data Token byte follows the 512-byte data. It must then be concluded with a 16-bit CRC byte. This 1+ 512 +2 = 515 bytes will be used to create a data packet. This data token consists of all bits except the LSB, which is assigned to 1(0xFE).

Fig. 8 Data format [9]

Fig. 8 the command to concern would seem to be 24, dummy bytes are sent until a clear R1 data is received, a dummy byte and then a token byte (11111110b) is being sent, this same 512 bytes of data will be sent following by two CRC's, and at last a data response has been received to tell if the data was viable, dummy data is continuously sent until the accurate response is received denoting the write has finalized, it would be obtaining any data other than zero. There isn't any SD card erase command this is done in hardware.

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

After receiving the data response, the SD card is ready for write; however, even before condition can be verified, the SD card must be de-asserted, timed eight times (1 byte), but then just reasserted. The preceding piece of the program may occasionally function, and that is frequently how datasheets are to be understood, but in order to commence the write process, the SD card must be deselected.

Whenever the SD card is busy, it pulls the output data lower (only if it is chosen), however when it pulls high, it indicates that it is ready. The program below is an additional technique to the one before although both produce the same result (ideally, we want 0xFF), this is actually the right procedure.WRITE BLOCK should be delivered again to write the next data block.

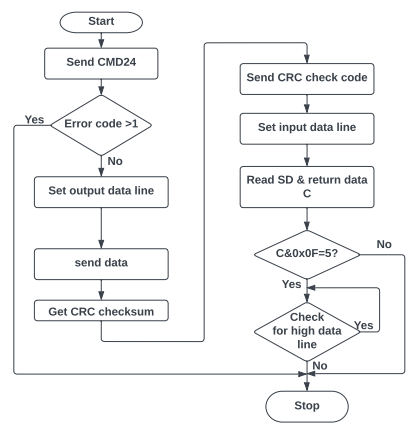


Fig. 9 Write data flow chart for SD card

*D. SD Card Read Data*

The data may be accessed from the SD card's 'Memory Core' using the instructions listed below.

READ\_BLOCK – Single block is read (512 bytes) from SD card.

***READ\_SINGLE\_BLOCK***

A block on an SD card is always viewed as a series of 512-byte memory addresses. When a block beginning at memory address of 2000 has to be read, use the READ SINGLE BLOCK command. The command packet must look just like:

0x51 is the first byte (command).

0x000007d0 - 2nd to 5th byte (argument) (This field must be set to zero even when there are no parameters for those other commands.)

(CRC) is the 6th byte with any random values.

7th byte NCR.

FPGA should get the R1 response once the command is sent. Every one of the bits should be zero. The FPGA may read data from the SD card when obtaining the zero-valued R1 response byte. SD card sends 512-byte data in response to every READ SINGLE BLOCK operation. cmd\_out<= {16h’FF\_51, address,8’hFF};

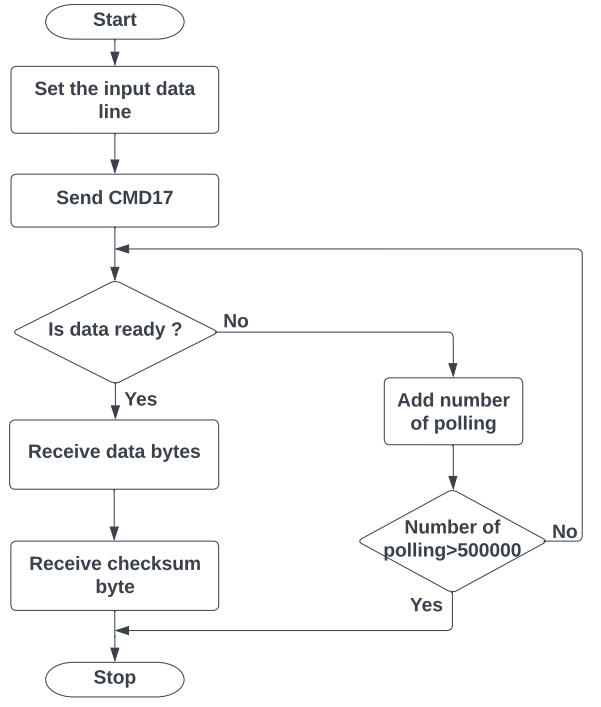
**

Fig. 10 Read data flow chart for SD card

*E. Control Working of SD card*

**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. File Format of FAT32*

Memory cores are used to store or write the FAT32 file system. At the start, there are specific sectors which are supported by Clusters. which is shown in Fig. 11.

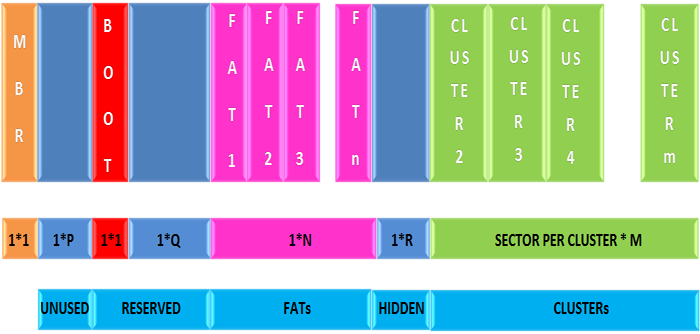


Fig. 11 Format of FATR32 file system in memory card [9]

MBR also known as Master Boot Record is the first sector. This comes after a big amount of idle sectors. Those sectors are supported as reserved sectors, the first of which is the BOOT sector, with reserved sectors being followed with FAT sectors. The amount of FAT sectors is determined by the file system's size. FAT sectors were followed by a few secure sectors, which are then followed by Clusters.

*G. Reading File from FAT32*

A file from the FAT32 file system may be read. It has been discovered that every procedure concludes with such a sector read. This sector read from the SD card's memory core may be accomplished alone by utilizing the SD Command Layer's READ SINGLE BLOCK command.

sdcard\_fat \_32\_read. V

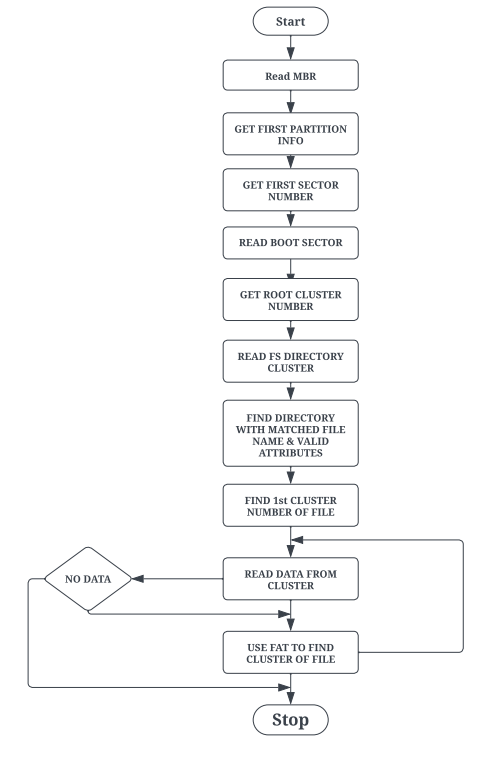


Fig. 12 Algorithm for Reading file from FAT32 file system [9]

FAT32 sectors feature of 32 bits that represent the number of clusters. Each 32 bit specifies a certain cluster. A cluster typically comprises 512 bytes, therefore the sector will include 128 cluster pointers. The following is the FAT32 File Allocation.

The following equation may be used to compute the number of the next cluster pointer within the FAT32.

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

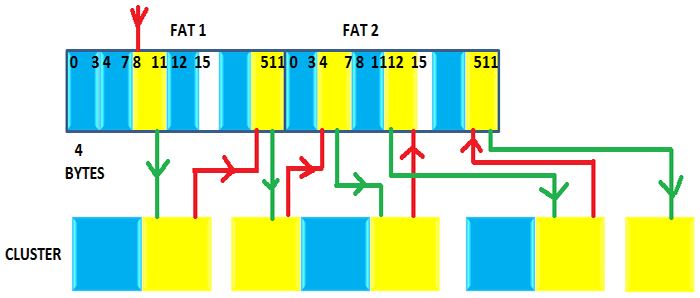


Fig. 13 Read Scrambler file across flash ‘Memory Core’ [9]

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 shows that the next cluster pointer which matches the present cluster is being sought, while the green line shows that the next cluster is being sought using the cluster thing about having 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 |

Fig. 14 Comparison bar chart between existing and our method

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 suggested the FPGA controller that can write and read SD cards in 4 - bit SD mode. The architecture suggests addressing each card in succession, but it is permissible. High amounts of memory are accessible.

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/).