**CHAPTER 1**

**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.

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.

**CHAPTER 2**

**LITERATURE SURVEY**

According to Dumitrel Catalin Costach [1] an FPGA controller reads and writes to an SD card using the SPI protocol. SPI transfer mode is necessary for the growth of basic storage systems. If we wish to utilize more SD cards, we must ensure that they all use the same clock, data, and chip select signals. This kind of memory access demands the use of several individual cards, yet it enables the storage of vast amounts of data.

According to Gul Munir Ujjan [4] introduced a fundamental hardware architecture based on FPGA and an integrated NIOS-II processor for processing SD cards in 4-bit SD mode. Using the Eclipse platform, software for the NIOS II processor is created. FAT-32 read-and-write instructions for a single block have been written and tested. These instructions can be executed in 1-bit SD mode. The read performance of SD cards with four bits is approximately 67% greater than that of SD cards with one bit. SD 4-bit mode is thus 80% faster than SD 1-bit mode during write operations. Improving overall performance may be a high concern, thus it is essential to provide a mechanism that does multi-block writes and reads and calculates 16-bit CRC write instructions quickly. This inevitably necessitates more expensive hardware resources. The proposed firmware would be beneficial for recording video in real-time, among other applications.

According to Pallavi Polsani [6] external A/D converters, D/A converters, and EEPROMs, serial synchronous communication between master and slave devices are used to connect the microcontroller to the other devices. There are two primary classifications of protocols. Priorities: 2) Inter-I2C Each protocol has been optimized for inter-IC communication on a board. SPI has become the predominant standard for delivering data streams at low to medium rates within and between processors. SPI (Serial Peripheral Interface) is a master-slave system that transmits data in bits and is highly configurable. For the verification and implementation of the SPI design, System Verilog was utilized. The functionality and code coverage of the system is validated. The whole RTL is created in Verilog for synthesis, whilst the System Verilog-designed and Spartan 3E-implemented verification architecture ensures quality.

According to J. Y. Qiang [5] it has been demonstrated that SPI bus is a sort of synchronous, full-duplex serial interface data bus line with a simple protocol and a high data transmission rate. FPGAs facilitate rapid device design and testing, making them a popular choice for parallel processing. Here, we shall discuss the creation and operation of SPI. The communication bus analyses and applies the operation sequence and four modes. An FPGA interface capability for an SPI bus is supplied by a module circuit. The SPI is designed using the hardware description language Verilog, while its waveforms are modeled using the vivado simulator. Simulation of waveform analysis demonstrates the feasibility of the method.

According to Omar Elkeelany [3] discovered it utilizing a programmable Field-Programmable Gate Array (FPGA) data extractor device that enables bidirectional SD card hardware architecture. SD card can read and write 5000 blocks per second, or 1.051 seconds, in total. In the previous attempt, 60 seconds were spent on the same block size. The variances stem from the researchers' selection for unique software-based techniques and finite-state machine designs. This proves the technological possibility of data transfer rates as high as 25 Mb/s. This letter offers a system for real-time SD card storage of data from SG actions at remote sites. Future work will address scaling issues, such as the impact of bigger SD cards and SD cards from various manufacturers.

According to Zinlin [2], it permits the rapid storing of essential data on an SD card. It then designs and develops an SD card reader system architecture based on FPGA to suit the requirements of SD memory card readers. Before being placed, the gadget underwent testing. The device's ability to read and write data to SD cards demonstrates that it satisfies the FPGA device requirement for outdoor garage devices. This needed the creation of an SD card reader that could be implemented on any number of FPGA-based devices.

**CHAPTER 3**

**SD CARD**

**3.1 SD CARD PRINCIPLE**

The SD card is consisting of two basic semiconductor sections, a ‘memory core’ and a ‘SD card controller’. The ‘memory core’ is the flash memory region where the actual data of the file is saved. When we format the SD card a file system will be written into this region. Hence this is the region where the file system exists. The ‘SD card controller’ helps to communicate the ‘memory core’ with the external devices like FPGAs, FPGA. It can respond to certain set of standard SD commands and read or write data from the memory core in for the external device.

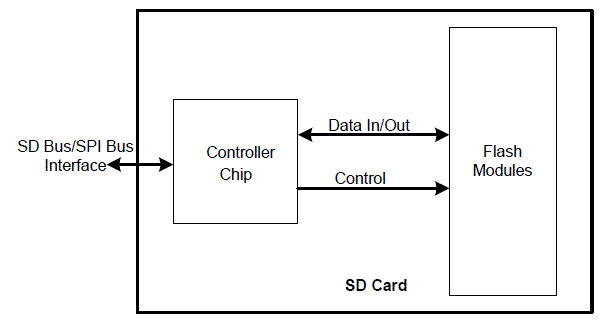


Fig 1 Block Diagram of SD Card.

The capacity of the ‘memory core’ is referred to as the size of the SD card. Other than the ‘memory core’ there are certain registers associated with the ‘SD card controller’. These registers store the status of the SD card. The contents of these registers are read only. The SD card can be interfaced with the FPGA using serial data bus. It can connect using ‘SD buses’ or ‘SPI bus’. The ‘SD bus’ is designed for high speed whereas the SPI bus can operate with much lower speed only. The FPGA can read or write data the memory core and read the registers using standard SD commands send through these serial buses.

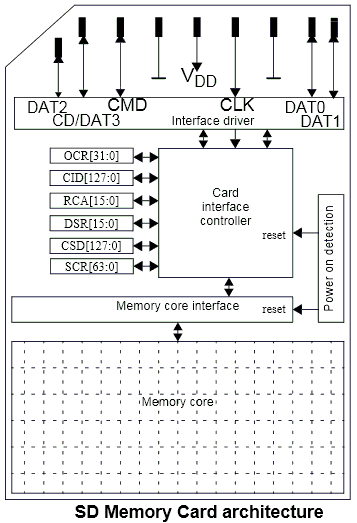


Fig. 2: Memory Architecture of SD Card

**3.2. MODE OF TRANSMISSION**

SD cards can transmit in one of three ways: the serial peripheral interface (SPI), which uses a different parallel input and output from 1-bit SD cards, as well as 1-bit SD cards' one-time transfer type and different command and data channels, and 4-bit SD modes, which use a wider parallel transmission bus and have additional pins and some reset pins. SD card mode provides faster and more reliable access. In 1-bit SD mode, the maximum transfer rate is 25 Mbps, whereas in 4-bit SD mode, the maximum transfer rate is 100 Mbps. Even if the data transfer rate is higher in 4-bit SD transfer mode, the system's complicated structure and timing make it challenging to build.

**3.3 BUS PROTOCOL OF SD CARD**

The idea behind SD communication is straightforward; it operates in a master-slave fashion. A master controller and many slave controllers connected through 3-6s lines are usual in such a setup. They are labelled as "CMD," "CLK," and "DAT0-3" (data lines).

Table 1. Pin functions of SD mode.

|  |  |
| --- | --- |
| **Name** | **Functions** |
| CLK | microcontroller uses this pin to send clock signal to SD card. |
| CMD | bidirectional pin for information and command transmission between microcontroller and SD card. |
| DAT0-3 | Four bidirectional pins are used for bulk data transfer data between microcontroller and SD card. |

**3.4 SYSTEM FILE OF SD CARD**

On the SD card, a FAT16 file is divided up into four portions. File Data Table (FDT), File Allocation Table (FAT), and Partition Boot Record (PBR) are the three tables that make up the File System (File Directory Table). The BPB (BIOS Parameter Record Blocks), the hard drive flag record book, the partition boot record code section, and the end Flag 55AA are the standard components of partitioned boot files.

**3.5 SD CARD COMMANDS**

SD internal memory requires a total of six bytes to execute an instruction. This consists of the one-byte command code, four bytes of command inputs, and the checksum bits. When transferring SD card instructions, the host system must always give 4 bytes of argument data, regardless of whether any parameters are included in the command. In this case, the SD card will just ignore the supplied value. Below Table 2. describes the commands of SD card and the functions of each command of the SD card.

Table 2. Commands of SD card for SPI mode

|  |  |
| --- | --- |
| **Commands** | **Functions** |
| CMD 0 | Software gets reset. |
| CMD 1 | Initialization starts. |
| CMD 3 | Request card send back the RCA address. |
| CMD 7 | Card entering the state. |
| CMD 9 | Read CSD registers. |
| CMD 10 | Read CID registers. |
| CMD 12 | Stop reading data |
| CMD 16 | Change of size in read/write. |
| CMD 17 | Read command for single command. |
| CMD 18 | Read command for multiple commands. |
| CMD 23 | Amount of block sent. |
| CMD 24 | Write in for single block. |
| CMD 25 | Write in for multiple blocks. |
| CMD 32 | Start erase block. |
| CMD 38 | Erase the command. |
| CMD 55 | ACMD <n> leading command. |
| CMD 58 | Read OCR. |

**CHAPTER 4**

**FPGA**

**4.1 FPGA PRINCIPLE**

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks. Although one-time programmable (OTP) FPGAs are available, the dominant types are SRAM based which can be reprogrammed as the design evolves.

The process of transforming the original system concept into an actual FPGA implementation that performs the required task is called functional design. 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. Fig 3. displays the FPGA development board.

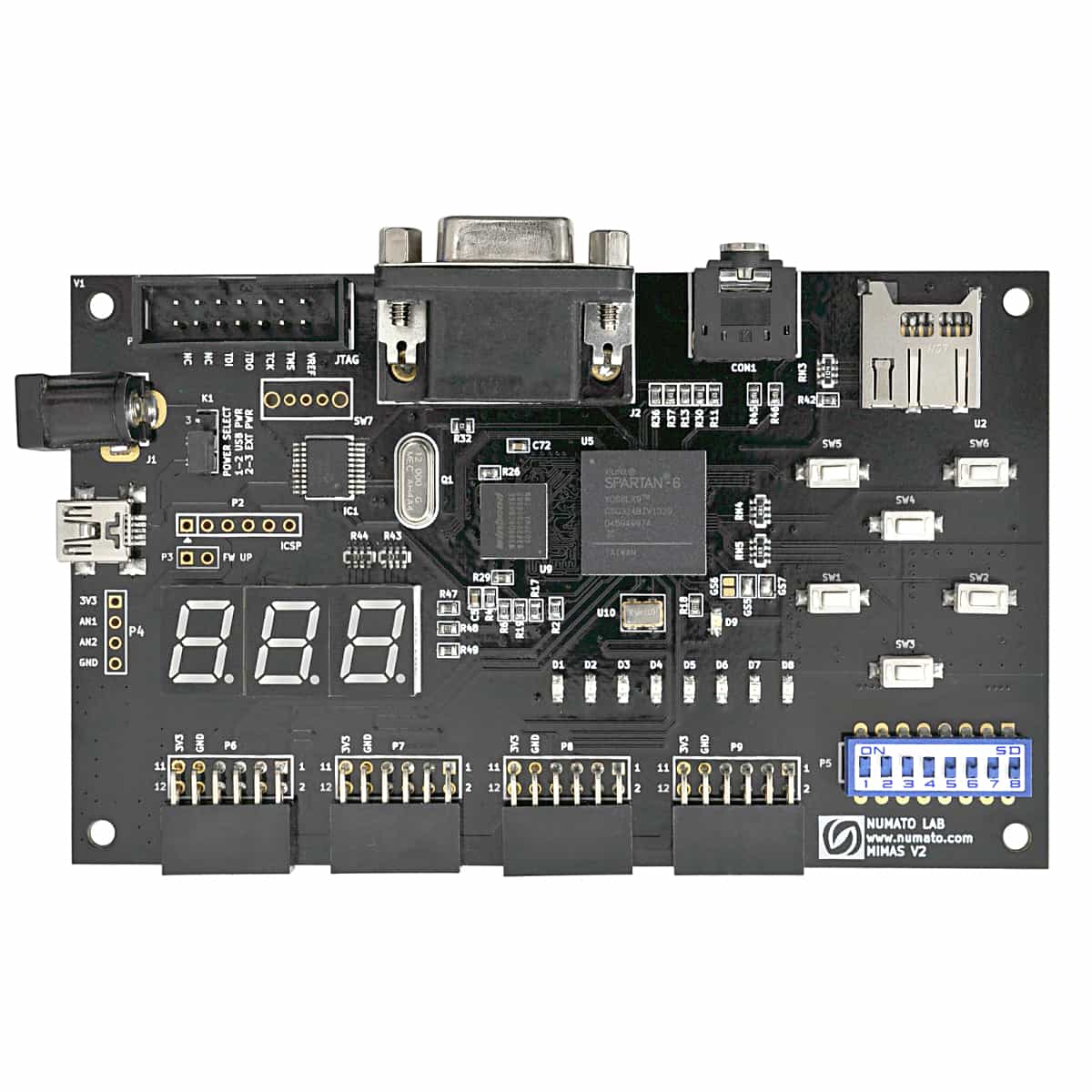


Fig. 3 FPGA Development Board

**4.2 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.

**CHAPTER 5**

**SPI**

**5.1 WORKING OF SPI**

The Serial Peripheral Interface (SPI) is a protocol utilized by many electrical devices. A card reader for SD or RFID cards and a wireless 2.4 GHz transceiver are examples of peripherals that utilize SPI to connect with a microcontroller. Sending and receiving any number of bits is possible so long as the flow is continuous. I2C and UART both transmit data in discrete packets with a defined number of bits. Start and stop criteria, which indicate the beginning and end of each packet, distort data transmission.

Master-slave is the relationship between SPI devices. The device in control is known as the master, while its subordinates (often sensors, displays, or RAM) are referred to as slaves. A straightforward SPI network consists of a single master and a single slave however a single master may supervise several slaves. Fig. 4 depicts the connection between the master and the slave.

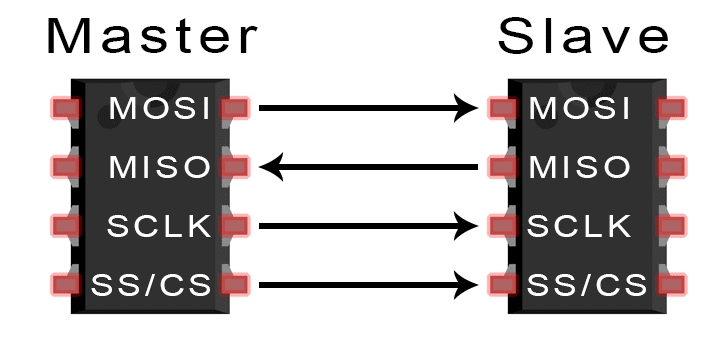


Fig 4 MOSI

**5.2 SERIAL VS PARALLEL COMMUNICATION**

Electronic devices sending bits of data through wires physically connected between devices. A bit is like a letter in a word, except instead of the 26 letters (in the English alphabet), a bit is binary and can only be a 1 or 0. Bits are transferred from one device to another by quick changes in voltage. In a system operating at 5 V, a 0 bit is communicated as a short pulse of 0 V, and a 1 bit is communicated by a short pulse of 5 V.

The bits of data can be transmitted either in parallel or serial form. In parallel communication, the bits of data are sent all at the same time, each through a separate wire. The following diagram shows the parallel transmission of the letter “C” in binary (01000011):

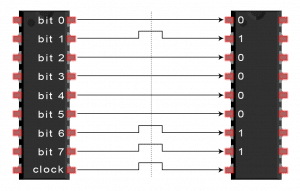


Fig 5 Bit Transmission in Parallel Communication

In serial communication, the bits are sent one by one through a single wire. The following diagram shows the serial transmission of the letter “C” in binary (01000011):

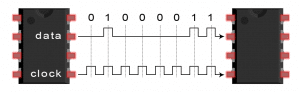


Fig 6 Bit Transmission in Serial Communication

**5.3 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.

**5.4 MISO and MOSI**

The master sends individual data bits to the slave over the MOSI connection. The master transmits data to the slave using the MOSI pin. The norm for data transmission between the master and the slave is most significant bit first. The slave sends the least significant piece of data back to the master.

**5.5 SPI DATA TRANSMISSION**

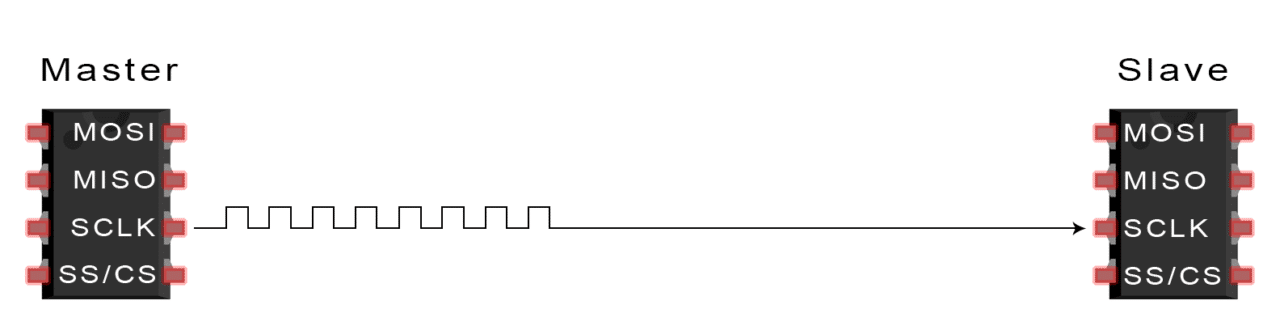


Fig 7 Clock Signal

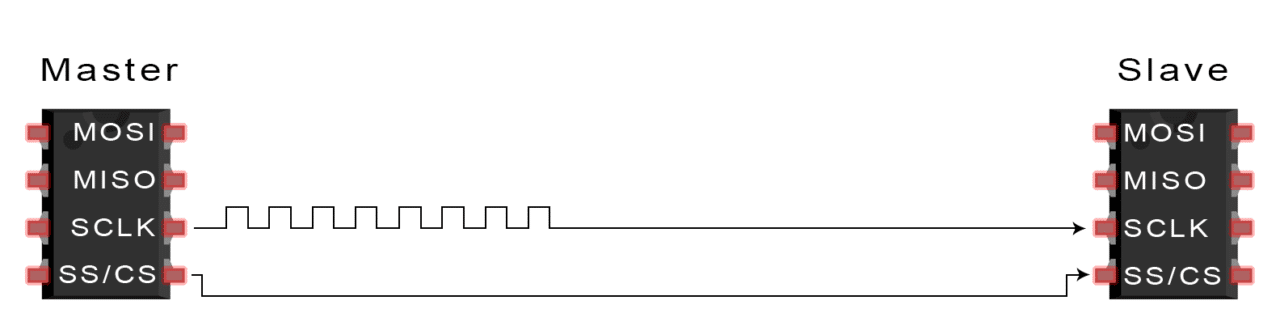


Fig 8 Slave Select Activation

Fig 9 Master to Slave Data Transfer

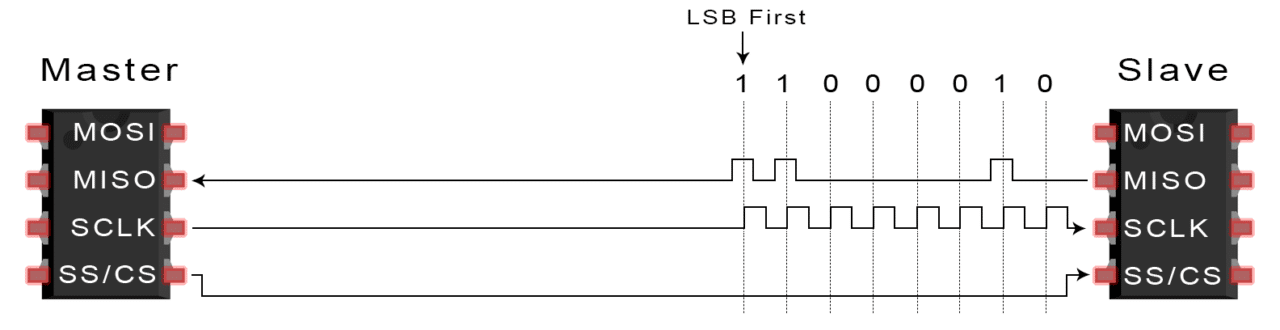
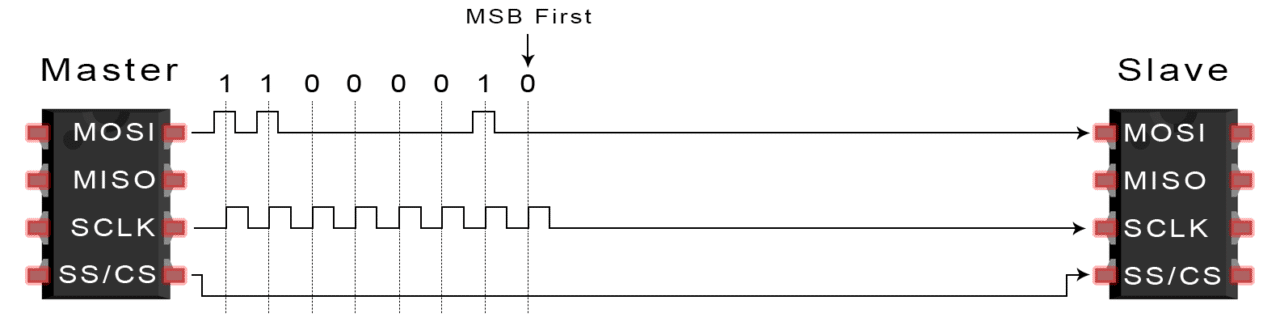


Fig 10 Slave to Master Data Transfer

The master generates the frequency of the clock. When the master pushes the SS/CS pins to a reduced voltage state, the slave is activated. The master transmits information bit by bit to the slave over the MOSI line. The slave inspected each received bit. When a response is required, the slave sends the data back to the master bit-by-bit over her MISO line. The master performs an analysis on the received bits.

**5.6 ADVANTAGES**

Since there is no beginning or end, information may be sent without interruption. The I2C bus does not require a sophisticated method of addressing that is slave-specific. The rate of data transfer is more rapid than I2C. By separating the MISO and MOSI channels, data transmission and reception may be performed concurrently.

**CHAPTER 6**

**WORKING SYSTEM**

**6.1 REQUIREMENTS**

**6.1.1 SOFTWARE REQUIRMENT**

* Xilinx ISE 14.7

**6.1.2 HARDWARE REQUIRMENTS**

* FPGA
* SD CARD
* SPI
* FAT32

**6.2 TYPES OF SD MODES**

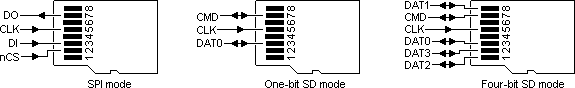


Fig 11 SPI mode pin diagram

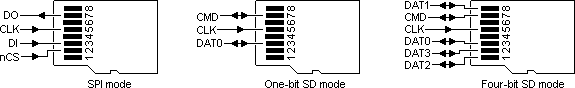


Fig 12 One-bit SDsmode pin diagram

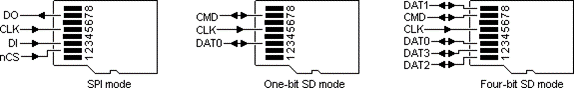


Fig 13 Four-bit SD mode pin diagram

Among these three modes from Fig 11, Fig 12, Fig 13 (SPI mode, one-bit SD mode and four-bit SD mode), the latter two are significantly slower than the former. The higher bit rate of four-bit SD mode allows for rapid and efficient command execution, including read/write operations.

Table 3 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 |

**6.3 CONTROL STRUCTURE OF FPGA**

Fig. 14 illustrates the recommended FPGA block architecture for 4-bit SD mode. Regarding the data pins, 4-bit SD mode requires only one clock and four data ones. The Clock Control module's principal purpose is information sharing. All functionality of 4-bit SD mode may be assured with the Command Control package. Due to the SD card's limited 512-byte memory location, two FIFO buffers are designed to temporarily store data during transmission and reception.

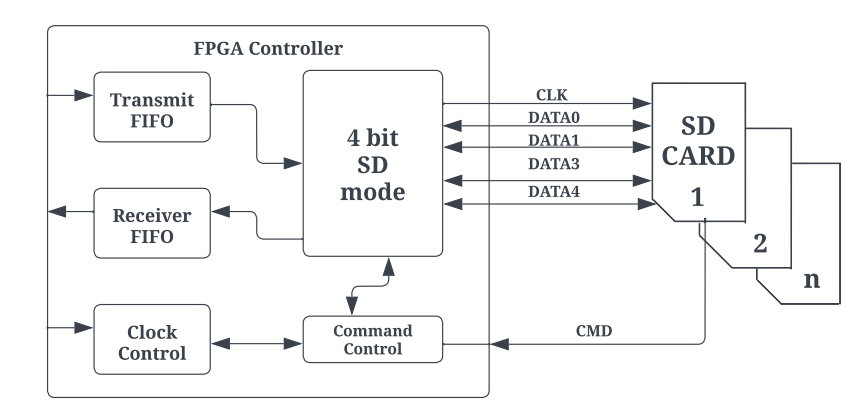


Fig 14 FPGA block diagram for 4-bit SD mode.

**6.4 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. We must enter the card choose pin. It will disable the card. SD must always be initialized with a minimum 76 - 160 pulses transmitted to the clock. SD lacks an internal clock primary source.

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 we 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 we need to understand is that the final byte we 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 15 illustrates the initialization of Secure Digital card.

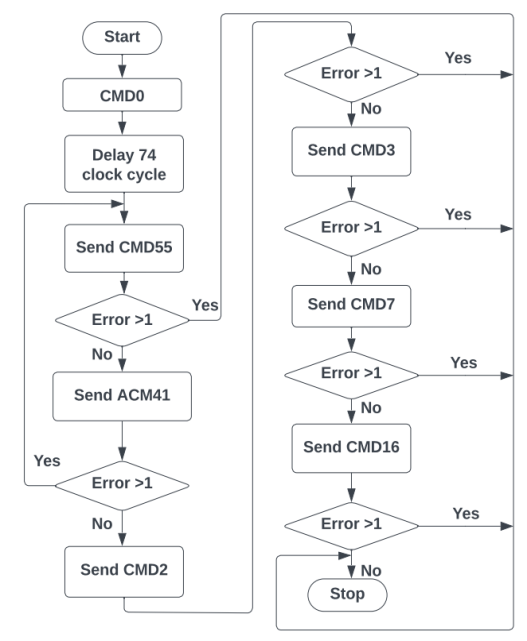
**

Fig 15 SD card initialization flow chart.

**6.5 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\_BLOCKs–Single blocks of data is being written (512 bytes).

***WRITE\_BLOCK***

A block on an SD card has been believed to be a sequence of 512 bytes of memory addresses for as long as anybody can remember. The WRITE BLOCK command is used to permanently store data in a memory block beginning at address 2000. For the command set to operate properly, it must appear as follows:

Everything begins at byte 0x18 (command). Second through fifth bytes (0x000007d0) (argument) (This option is needed to be 0, even when the other commands have no further arguments. (CRC) represents the random values included in the sixth byte. Seven-byte NCR. After receiving the command from the FPGA, the R1 should respond. Every bit must be zero. With the R1 response byte set to zero, the FPGA is able to transmit the data for writing to the SD card. The minimal data length should be 512 bytes. This is true even if the length of the real data is shorter. Immediately following the 512 bytes of data is the Data Token byte. Next, a 16-bit CRC byte must be utilized for the final verification. The total number of bytes in the data packet will be 515, which is equivalent to 1 plus 512 plus 2. The least significant bit (LSB) is set to 1 in this informative token (0xFE).

The command to be concerned with is 24, and until a clear R1 data is received, dummy bytes are sent; a dummy byte is followed by a token byte (11111110b); the same 512 bytes of data are sent; two CRCs are sent to determine if the data is valid; and finally, a data response is received to determine if the data is viable. This is accomplished in hardware, not via an SD card erase command.

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

After receiving the data response, the SD card may be written to; however, before the card's status can be verified, it must be de-asserted eight times (1 byte) and then reasserted. While the preceding code snippet may occasionally function and is frequently how datasheets are to be understood, the SD card must be deselected prior to beginning the writing procedure. When the SD card is being used for something else, the output data will be reduced (if this option is set), but when it is no longer in use, the output data will be increased to indicate that it is accessible. Fig. 16 illustrates write data of Secure Digital Card.

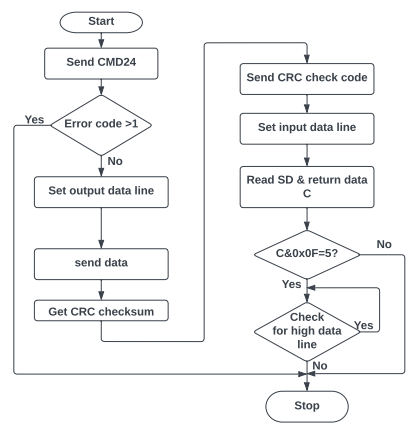


Fig 16 Write data flow chart for SD card.

**6.6 READ DATA FOR SD CARD**

We can retrieve the information saved in the "Memory Core" of SD card by following the instructions listed below.

A single block of data (512 bytes) is read from the SD card in READ BLOCK mode.

***READ\_SINGLE\_BLOCK***

Each SD card block is represented by a 512-byte memory address sequence. The READ SINGLE BLOCK command is used to get a single block of data beginning at the given memory address. In this instance, 2000 is the beginning address. The command packet format must be as follows:

When the time comes to record the succeeding data block.

0x51 is the value of the first byte (command).

(The argument) positions 0x000007d0 through 0x000007ff (This field must be set to zero even when there are no parameters for those other commands.) (CRC) represents the random values included in the sixth byte. Seven-byte NCR. The response from R1 should reach FPGA immediately following the instruction. All bits should be set to zero. The FPGA may get information from the SD card if the R1 response byte is zero. Each READ SINGLE BLOCK request from an SD card returns 512 bytes in response. Fig. 17 illustrates read data of Secure Digital Card.

cmd\_out<= {16h’FF\_51, address,8’hFF};

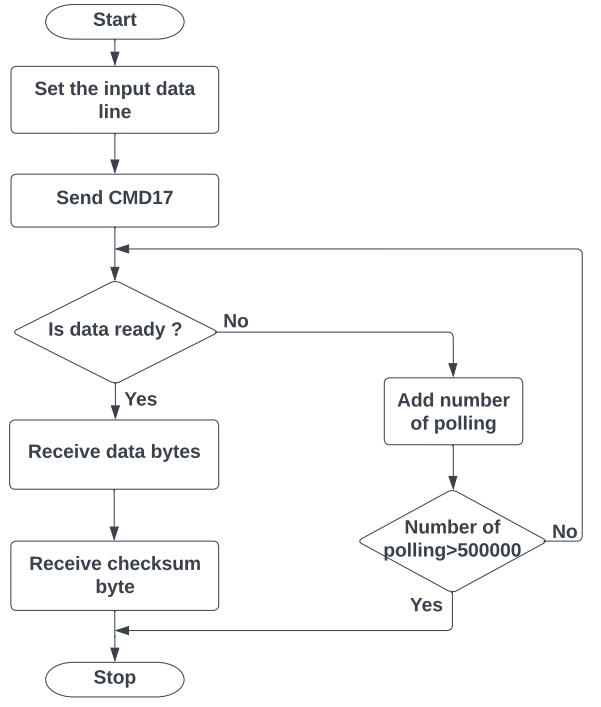
**

Fig 17 Read data flow chart for SD card.

**6.7 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

**6.8 FILE FORMAT of FAT32**

Data is read from and written to memory modules using FAT32. Initially, clusters concentrate on a small number of sectors. It is displayed in Fig. 18.

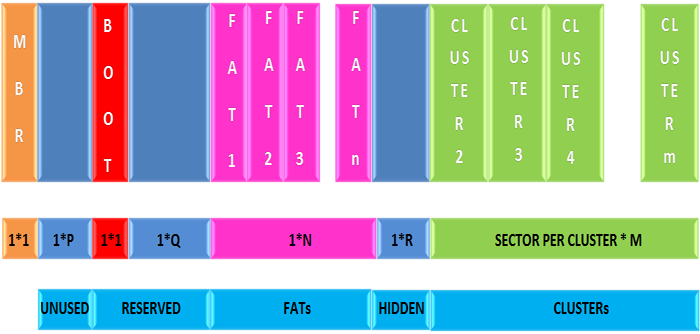


Fig 18 Format of FAT32.

The first and foremost sector is the MBR which is known as Master Boot Record. This comes after a lengthy time in which many sectors remained inactive. These sectors are designated as reserved ones; the BOOT sector comes first, followed by the FAT sector. The count of FAT sectors is determined by the file system size. Clusters followed FAT sectors and before insecure ones.

**6.9 READING FILE FROM FAT32**

Fig. 19 illustrates that it is able to read FAT32-formatted files. This sector read has been discovered to be the final phase of every operation. We can use the READ\_SINGLE\_BLOCK command supplied by the SD Command Layer to read only this one sector from the memory core of the SD card.

sdcard\_fat \_32\_read. V

From Fig. 10FAT32 sector cluster counts are expressed by 32 bits. Every cluster is represented by a 32-bit set. If each cluster consists of 512 bytes, then the sector will have 128 cluster pointers. This section describes how space is allotted to files in a FAT32 file system.

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).

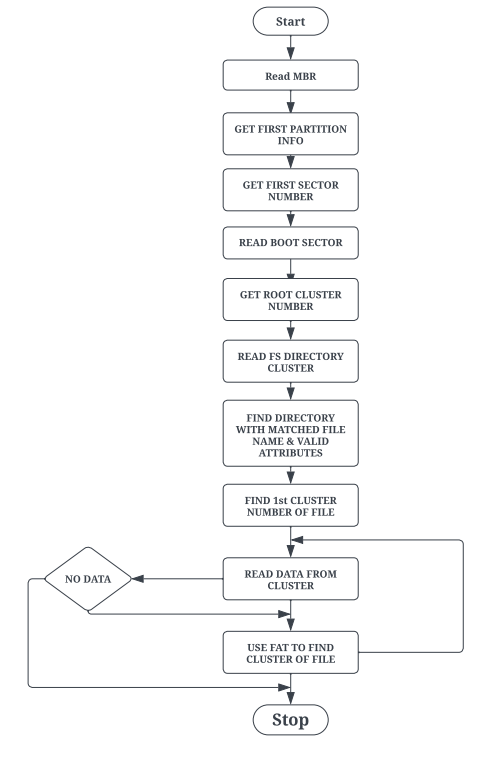


Fig 19 FAT32 Algorithm for reading.

From Fig. 20 the yellow-coloured box indicates the cluster that contain the data for each file in FAT32 and the respective cluster pointers. The red-coloured line shows that the next cluster pointer which matches the present cluster is being sought, while the green-coloured line shows that the next cluster is being sought using the cluster thing about having in the FAT32 clusters pointer.

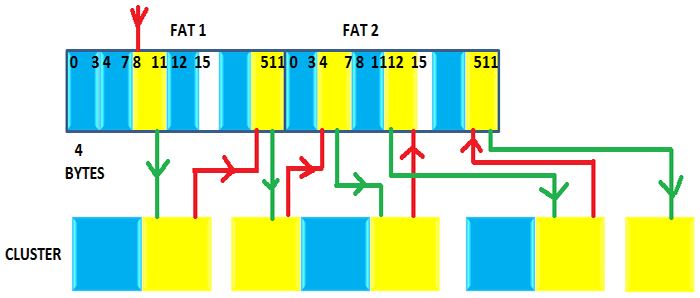


Fig 20 Read Scrambler file in flash ‘Memory Core’.

**6.10 PROJECT SOURCE CODE**

Sdcard\_controller.v

Sdcard\_fat32.v

Ram.v (512)

FAT\_top.v

VGA\_driver.v

**CHAPTER 7**

**RESULTS**

Table 4 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 | 1025 | 579 | 10.5 ns |

Comparing the specifications of the Spartan 6 FPGA (530 Slice Registers, 916 LUT's, 464 Flip Flops, 5.35 ns clock period, 186.925 MHz frequency) and the Artix-7 (512 Slice Registers, 653, LUT's, 413 Flip Flops, 3.254 ns clock period, 307.342 MHz frequency) demonstrates that our proposed results provide better resource utilization than existing methods.

**CHAPTER 8**

**CONCLUSION**

Thus, we conclude our proposed work that an FPGA controller allows SD card writing and reading with Strontium 4-bit SD mode using Spartan 6 FPGA and Artix-7 which was executed in Xilinx ISE software tool. 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.

**8.1 FUTURE SCOPE**

* For future work, in the implementation of FPGA-based classification systems, it is proposed to change the SD memory for a Double Data Rate 3 Synchronous Dynamic Random-Access (DDR3-SDRAM) memory, and test with different configuration to reduce the access time; in the algorithmic part it is proposed to reduce the extraction time of the characteristics using time-frequency characteristics.

**CHAPTER 9**

**REFERENCE**

[1] D. C. Costache, L. A. Perişoară and A. Florescu, "FPGA Implementation of a SD Card Controller using SPI communication," 2020 12th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), 2020, pp. 1-4.

[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] O. Elkeelany and V. S. Todakar, "Data Archival to SD Card Via Hardware Description Language," in 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, DaNang, 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] N Nandhagopal, S Navaneethan, V Nivedita, A Parimala, D Valluru “Human Eye Pupil Detection System for Different IRIS Images” Journal of Computational and Theoretical Nanoscience volume 18, issue no 4, page no 1239-1242, 2021.

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

[10] V. Asanza, A. Constantine, S. Valarezo and E. Peláez, "Implementation of a Classification System of EEG Signals Based on FPGA," 2020 Seventh International Conference on eDemocracy & eGovernment (ICEDEG), 2020, pp. 87-92.

[11] S Navaneethan, N Nandhagopal, V Nivedita “An FPGA-based real-time human eye pupil detection system using E2V smart camera”, Journal of Computational and Theoretical Nanoscience 16 (2), 649-654 2019.

[12] Navaneethan S, Nandhagopal N “RE-PUPIL: resource efficient pupil detection system using the technique of average black pixel density” Sadhana vol.46 issue no. 3 2021, Springer India

[13] N Nandhagopal, S Navaneethan, V Nivedita, A Parimala, D Valluru “Human Eye Pupil Detection System for Different IRIS Database Images” Journal of Computational and Theoretical Nanoscience volume 18, issue no 4, page no 1239-1242, 2021.

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

[15] M. K. Birla, "FPGA Based Reconfigurable Platform for Complex Image Processing," 2006 IEEE International Conference on Electro/Information Technology, 2006, pp. 204-209.