SD Card Host Controller Design and Verification

M.Tech Project Report II

Semester: Spring, 2020-21

Submitted by

Name: Arghya Kamal Dey

Roll No. :193070053

Integrated Circuits and Systems

Under Guidance of:

Prof. Madhav P. Desai

Department of Electrical Engineering

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

Powai, Mumbai-40076

June 2021

Dissertation Approval

The report entitled

"**SD Card Host Controller Design and Verification**"

by

Arghya Kamal Dey

(Roll No. 193070053)

is approved for the degree of

Master of Technology with Integrated Circuit and System specialization

_____ _____

(Examiner) (Chairperson)

_____

(Supervisor)

Date:

Place:

Declaration

I declare that this written submission represents my own ideas in my own words and where other's ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misinterpreted or fabricated or falsified any idea/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

Arghya Kamal Dey

Roll No 193070053

Date :

Acknowledgments

I would like acknowledge my supervisor, Prof. Madhav P. Desai for his guidance and mentorship throuhgout the period, which helped me in the persuit of excellence.

Abstract

We want to build a system for an SD card around the AJIT core. This system will help us integrating a new memory system with the already existing system for the AJIT core. An SD card can be used to transfer file system in much faster speed and can have higher capacity. The aim of this project is to build a system which can transfer data at a high speed with the core. The memory systems currently used for storing data like flash memory and DRAM The design choices are made such that the system gives the required performance as well as high capacity of memory. There are different modes of communication with an SD card. For communicating with an SD card an SD host controller needs to be designed. The design specifications and the internals of the host controller are discussed with block diagrams. The design is made in multiple steps which is shown in this report. The design needs a verification plan which will verify the functionality of the SD card host controller.

Contents

List of Figures

List of Tables

1. Introduction

1.1 Introduction to the Project

An SD card gives that flexibility in storing file systems in a non-volatile manner. Also, a much higher storage capacity can be achieved. The performance and speed of the SD card can be much higher also. Therefore the SD card system is being integrated into the AJIT processor core system.

The SD card needs a host controller to communicate with the processor. SD card host controllers are specified by the SD Association. The design methodology is shown in this report.

The version which we are going to follow and the reason behind it is discussed in the report. The specification of the SD card and the host controller is discussed here in the report. All the design decisions made are explained. The design steps are shown in detail. The design needs to verify at each step. The functioning needs to be verified using a testbench. There will be logical view verification and physical view verification of the SD card host controller.

In logical view, the algorithm of the SD host controller is verified. The physical view will test the performance of the sd card host controller. The verification procedure is discussed here in detail.

At an instance of time how much of the hardware is being used is known as utilization. The design has been optimized for better utilization. The procedure for increasing the utilization is also described in this report.

2. SD Card Host Controller Overview

2.1 SD Host Architecture

The SD Host controller is the hardware interface that helps connect the SD card with the host software. SD Host controller being implemented here follows SD Host controller specification version 3.00. This helps in developing an SD host driver for Linux without looking at the specific host controller specification.

The host driver cannot directly communicate with the SD card. It can only read or write the SD host controller registers. That is why it is important to standardize the registers. The driver here works both as sd host bus driver and host controller driver. Therefore driver is made specifically for ajit peripheral bus which can support the data transfer.

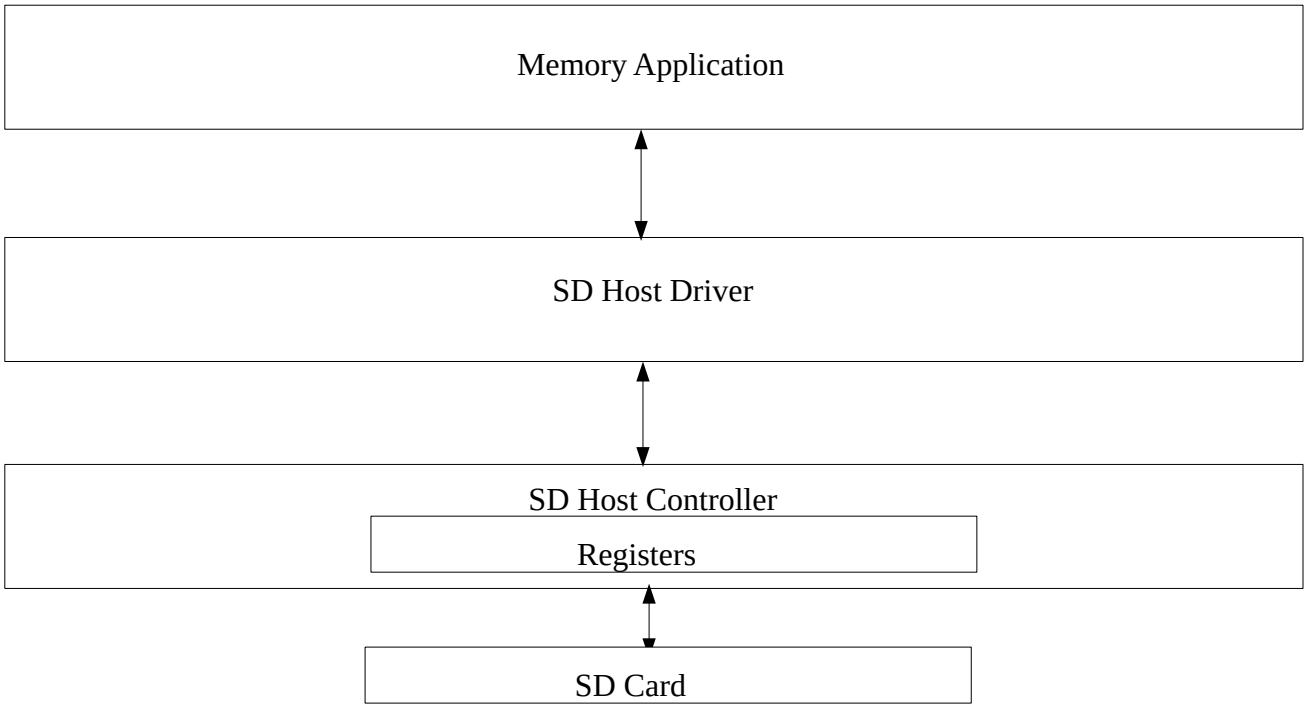

Figure 2.1 : Host hardware and driver architecture

2.2 Host Controller Features

- Meets SD standard specifications 3.0
- Supports SD 4-bit mode only.
- Initializes the SD card at default speed of 400KHz.The data transfer can be at normal clock of 25 Mhz or at higher speed clock of 50Mhz.

- The data transfer rate can be at default speed of 12.5 MBps or at higher speed of 25 MBps.
- It has a read or write buffer of 512 byte size for processor operated read or write.
- Supports single block data write and data read.
- Supports multiple block write and multiple block read.
- Read and write operations are done in the size of blocks only.
- The size of blocks is fixed to 512 bytes.
- Single interrupt line is used.
- AFB protocol is used to connect the processor with the host controller.

2.3 Host Controller Interfaces

2.3.1 Clock, Reset Interfaces

• A single clock signal Clk comes as input. The controller is a positive edge-triggered.

• A single active-high synchronous reset signal as input.

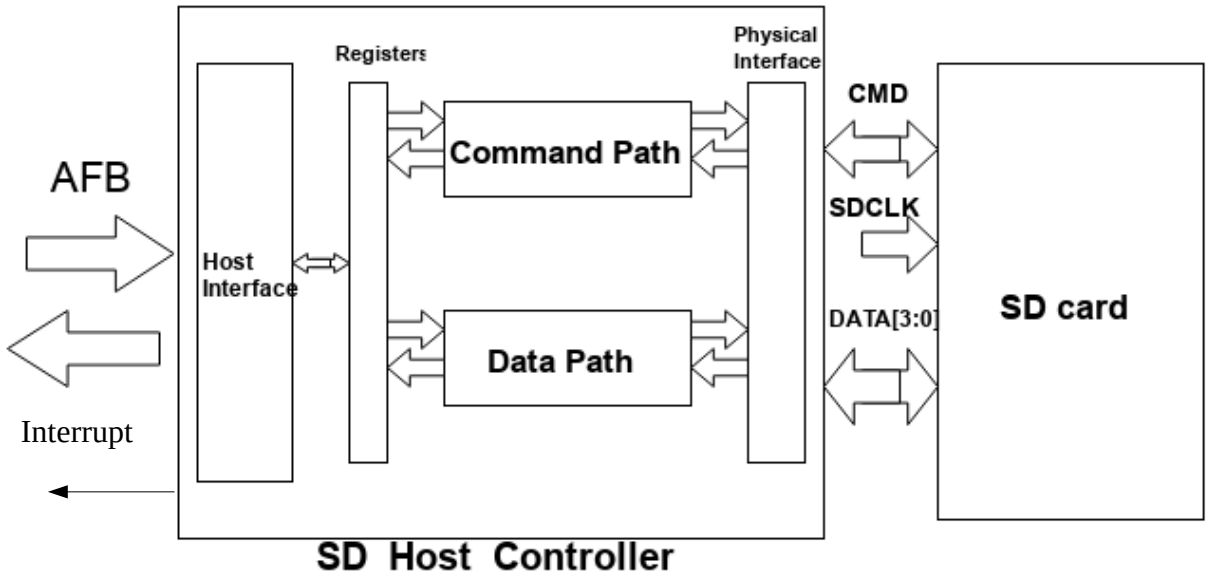

Figure 2.2 : Block Diagram of Host Controller

2.3.2 Processor Side Interface

On the processor side, there is peripheral bus interface.

1. **Request FIFO Interface**

peripheral_bridge_to_sdhc_request_data : IN (63: 0);

peripheral_bridge_to_sdhc_request_req: IN(0:0);

peripheral_bridge_to_sdhc_request_ack : OUT(0:0) ;

peripheral_bridge_to_sdhc_request_data data fields are given below:

63	62:59	58:56	55:32	31:0
rwbar	bytemask	reserved	address	data

Table 2.1 : Request data break-up

2. **Response FIFO interface**

sdhc_to_peripheral_bridge_response_data: OUT (31:0);

sdhc_to_peripheral_bridge_response_req : IN(0:0);

sdhc_to_peripheral_bridge_response_ack : OUT(0:0);

3. Interrupt Signal

SDHC_to_IRC_INT : OUT(0:0);

2.3.3 SD Card Bus Interface

SD_CLK: OUT (0:0) :Provides clock to the SD card.

CMD: INOUT(0:0) :A bidirectional wire that is used to send commands serially to the card and receive their corresponding responses from the addressed card to the host.

DAT : INOUT(3:0) : A bidirectional bus used to transmit data from the host to the card and vice versa. The DAT signal operates in push-pull mode.

2.4 SD Host Standard Registers

Following are the registers of the SD device as described in the SD host controller version 3.00.

Sl No.	Registers	Width	Address Offset
1	Block Size Register	16	0x4
2	Block Count Register	16	0x6

3	Argument Register	32	0x8
4	Transfer Mode Register	16	0xC
5	Command Register	16	0xE
6	Response0 Register	32	0x10
7	Response2 Register	32	0x14
8	Response4 Register	32	0x18
9	Response6 Register	32	0x1C
10	Buffer Data Port Register	32	0x20
11	Present State Register	32	0x24
12	Host Control Register	8	0x28
13	Clock Control Register	16	0x2C
14	Software Reset Register	8	0x2F
15	Normal Interrupt Status Register	16	0x30
16	Error Interrupt Status Register	16	0x32
17	Normal Interrupt Status Enable Register	16	0x34
18	Error Interrupt Status Enable Register	16	0x36
19	Normal Interrupt Signal Enable Register	16	0x38
20	Error Interrupt Signal Enable Register	16	0x3A
21	Capabalities Register	64	0x40
22	Host Controller Version Register	16	0xFE

Table 2.2 : Register Set in Host controller

3. SD Card Host Controller Logical View

The Logical view is the part of the design where the algorithm of the design is written. In this part of the design, the host controller is described as an algorithm in the Aa language and all the interfaces are FIFOs only.

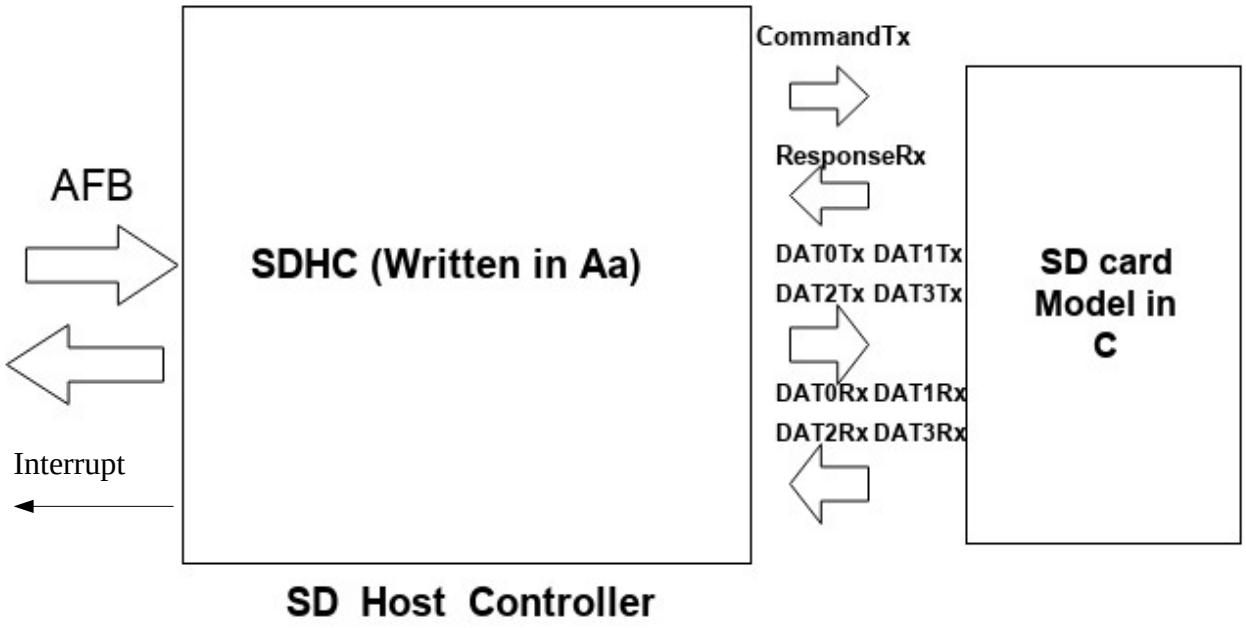

Figure 3.1 : Block Diagram of Host Controller in logical view

The SD host controller design started with the logical view where the logic of the design is verified. The Aa code written for the design is tested with an SD card model written in C and a testbench written in C. The design in Aa is converted to VHDL using the Ahir tool which gives us a preliminary idea about the performance.

The SD Host controller is designed in different modules. These modules are described below.

3.1 SDHC Daemon

The daemon is a module that runs forever. Here this SDHC daemon module is treated as the top module which is always running. This module is responsible for reading from the request FIFO peripheral_bridge_to_sdhc. It waits till there is some data in the FIFO. After reading the data it is

decoded to check whether the processor wants to read or write. Then the address of the register is used to write into that register or read from that register through the sdhc_to_peripheral_bridge response FIFO. In case write data zero is sent through the response FIFO. Also, it checks the interrupt-related registers to find whether there is an interrupt to be generated. If an interrupt is to be generated the module makes SDHC_to_IRC_INT line high otherwise low.

According to the SD host controller specification command is to be generated when the processor writes into the most significant byte of the command register. This module calls the command generator module when required. It also calls all the other modules as and when required.

3.2 Command Generator

The command generation in the host controller follows a specific sequence. First, it has to read the contents of the argument register. Then the command index is read from the command register. The format for the command is shown in Table 3.1. Then the CRC7 is calculated for the first 40 bits where the start bit and transmission bit are also added. When the crc7 is calculated it is added with the first 40 bits and an end bit '1' is added at the end. It sends the command through the FIFO named commandTx.

After sending the command it may have to wait for the response depending on the conditions set on command registers response type bits. Once a response is read from the responseRx FIFO it checks the correctness of the crc7 received and changes the interrupt status register values according to that. The crc7 generator polynomial here is $x^7 + x^3 + 1$. Then it stores the response in the response register. It also sets the command complete bit at the end in the normal interrupt status register. It also has to change the bits in the present state register according to the state after reading the current state from the present state register.

Bit position	47	46	[45:40]	[39:8]	[7:1]	0
Width(bits)	1	1	6	32	7	1
Value	'0'	'1'	X	X	X	'1'
Description	Start Bit	Transmission Bit	Command Index	Argument	CRC7	End Bit

Table 3.1 : Command Format

3.3 Data Transfer

The data transfer occurs through a buffer. The data should be sent in blocks of 512 bytes. The processor needs to divide the data into 32 bits and write to the data buffer port register. The buffer is written into from the data buffer port register. For 4 -bit SD-mode the data needs to be broken into the format shown below in Fig 3.2.

For data blocks, CRC16 is used. The generator polynomial for CRC16 is $x^{16} + x^{12} + x^5 + 1$. The data is transferred through the four FIFOs DAT0Tx, DAT1Tx, DAT2Tx, DAT3Tx. Then transfer complete interrupt bit is set.

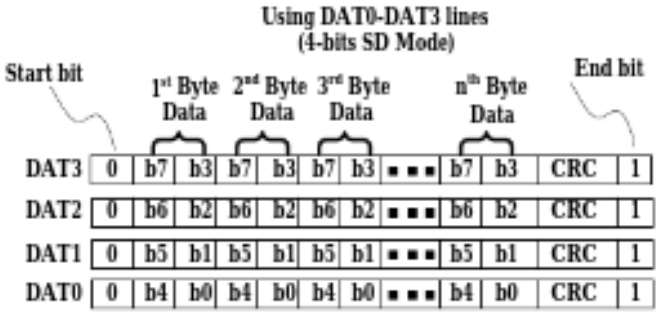

Figure 3.2 : Byte-width data-packet format when using 4-bit SD mode

3.4 Data Recieve

The data received from the SD card follows the same format as shown in Fig 3.2. The data is received through the 4 FIFOs DAT0Rx, DAT1Rx, DAT2Rx, DAT3Rx. Once a data block is received it has to check the CRC of the four lines as well as the end bits. According to the condition checks normal interrupt and error interrupt status bits are set or cleared. The data is then stored in the buffer. The data can be read back by the processor from the data buffer port register as 32 bits at once. For multiple block read the data needs to be read by the processor before the next block can be read from the SD card.

3.5 Register Write

There are 6 different types of register bits possible in the host controller register set. They are ROC, RO, RWAC, RW1C, RW ,and HwInit. The ROC is the read-only type of bit which means this is write protected from the processor. But ROC type bits can be cleared when software reset is set. The RO type bits are the same as ROC. The only difference is that it does not change with the software reset. When an RWAC bit is set by the host driver it requests the host controller to do some operation and at the end of the operation that bit will be cleared. The RW1C is set by the host controller only. Writing a 1 to these bits implies that the bit needs to be cleared. The RW is the normal read and write bit which is read and written by both the host controller and the driver. The HwInit is the bit that is hardware initialized and cannot be changed. The register write module protects writing into RO, ROC, HwInit bits. It also clears RW1C bits when 1 is written into them.

3.6 Software Reset

There are several types of software reset controlled by the values in the software reset register. It can be a ‘software reset for all’, where all the register bits except HwInit are cleared. In the case of ‘software reset for CMD line’, only registers and bits related to the command line are cleared. In the case of

‘software reset for DAT line’, only registers and bits related to data lines are cleared. A different module is required here so that the reset does not affect the ROC and HwInit bits unnecessarily.

All these modules together complete the SD host controller design in the logic view which can be verified using the SD card model in C.

4. SD Card Host Controller Physical View

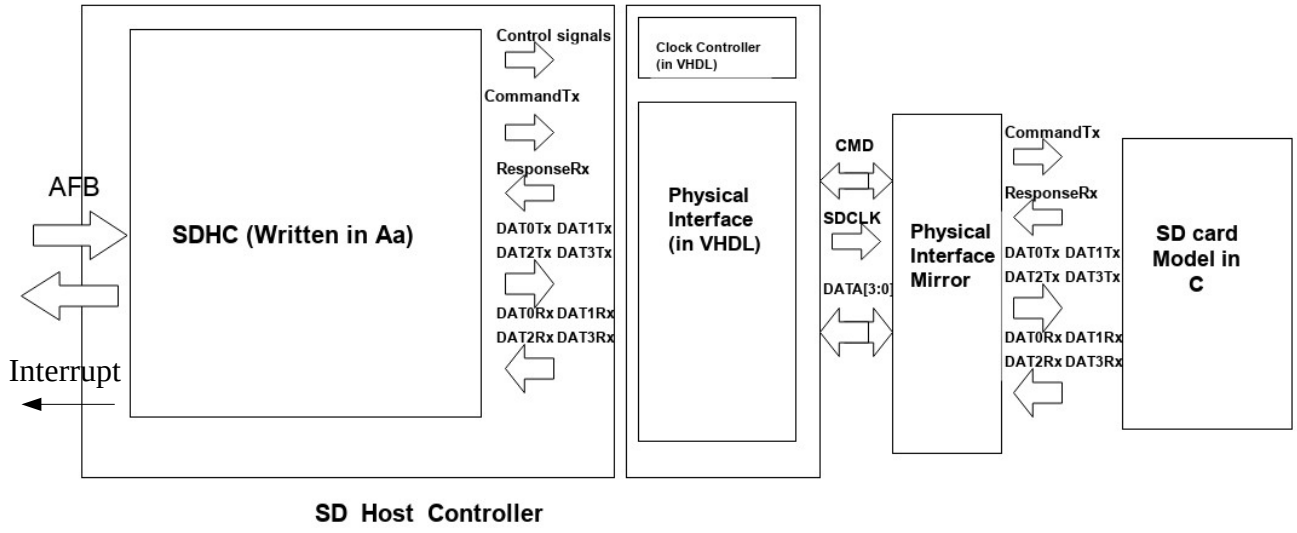

Figure 4.1 : Block Diagram of Host Controller in physical view

In the logical view, there is no clock present in the design. Also, all the interfaces are FIFOs. But in the physical view, the interfaces are the same as the physical implementation.

For physical view, some more blocks are added with the SD host controller in the logical view. These are written in VHDL and using all these blocks as components a top-level module is designed which is the final SD host controller.

The blocks which are written in VHDL to complete the phsyical view are described as follows.

4.1 Ahir System

This is the block that is generated by the ahir tool from the design written in Aa. This is the most important block for the SD host controller. It retains all the features from the Aa code but it is concurrent and not sequential. In physical some more control signals are generated from the Aa code. These signals help in controlling different operations in the physical interface. The signals like output enable help in the tristate logic and frequency select helps in selecting the sd clock frequency etc. More details about the signals are given in the following modules.

4.2 Clock Controller

The clock controller module can generate any frequency less than the base frequency of the host controller. The base frequency in this host controller is 100Mhz. The frequency select signal is a 10-bit divider that is generated from the clock control register. The resulting output is the SDCLK signal which goes to the SD card. The SDCLK is calculated by dividing the base frequency by the divider value. The resulted sd clock always has 50% duty cycle. The maximum clock that can be generated is 50Mhz. The default frequency is kept as 400 Khz. This is the frequency at which the sd card must be initialized. Then the frequency should be changed by the host driver to a higher frequency for data transfer. Before changing the clock frequency the sd clock enables bit should be cleared by the driver. When the sd clock is disabled the sd clock enable signal which is one of the control signals disables the sd clock.

4.3 Physical Transmitter

The transmitter block written in VHDL takes input from the FIFO and outputs a bitstream synchronized with the sd clock. Every FIFO has three signals which are data, req ,and ack. When the FIFO has some data to be read the ack signal is raised and when the environment is ready to read the data it raises the req signal. If at a positive clock edge both req and ack are high then the data is read.

In this block, req is generated keeping it synchronized with the sd clock signal. Because the bit tream generated will be read in every negative edge of the sd clock by the sd card. The components of this block is used to generate bitstreams for every output pipe from the ahir system.

4.4 Physical Receiver

This block does the vice-versa of the transmitter block. It takes bitstream as the input and stores the data into the FIFOs. For receiver blocks. there are response receiver and data receiver. The data receiver uses a counter to store a block of data into a FIFO. In case of DAT0 line there can also busy signal sent from the sd card. For the busy signal, a different mechanism is used in the receiver block where the Aa code sends a busy check signal from which the receiver can understand that the counter will not work in this case and has to wait indefinitely till the signal is raised high which implies that the card is not busy anymore. Therefore a separate component is used for storing into responseRx, DAT0Rx and the rest of the data lines. For multiple block, data read the FIFO needs to have a large depth otherwise there is possibilty of data loss. This block is also synchronized with sd clock so that it can sample the bitstream at the right clock edges.

4.5 Bidirectional

This block generates the inout or tristated signal from two signals. For this, an output enable ‘oe’ is generated from the ahir system. When the oe is high the tristated logic sends the input to the tristate signal. Otherwise it will receive the bitstream from the tristate signal and send it to the output signal.

There is a total of 5 such tristate signals in the design. The CMD line and the four data lines. In one component the outgoing command and the incoming response are tristated. In the rest of the four components, the data transmission outputs and data receiver inputs are tristated.

4.6 Physical Interface Mirror Logic

For verifying the physical view we need to convert the tristate logic back to the FIFOs. This helps in connecting the sd card model written in C to the host controller design. Therefore there is another VHDL code in this block which mirrors whatever the physical blocks described so far. Therefore this block again tristates the bitstream into individual input and output. Then the bitstream is again stored into the respective FIFOs.

The details of the verification methodology will be described in the next chapter.

5. Verification Methodology

The verification is done in two stages. First, the logical view is verified using the sd card model written in C along with the testbench written in C which imitates the host driver. Then the physical view is verified to verify the tristate logic and other logic written in VHDL. The methodology is described below.

5.1 Logical View Verification

For the logical view verification, a testbench is written in C and all the interfaces are FIFOs except a signal for the interrupt. The sd card model receives the commands from the commandTx FIFO and send the response accordingly through the responseRx FIFO. The testbench carries the initialization sequence. So all the commands are sent in that sequence and then data is sent. Both single block data and multiple block data is sent and received. A block of data is sent and stored in the sd card model first and then it is read back. After that, it is checked whether the received data is the same as the sent data. The same procedure is followed for both single and multiple block read. The test setup for the logical view is shown in Fig 3.1.

5.2 Physical View Verification

In physical view, the same testbench can be used with a few changes being made. The sd card model will remain the same. In physical view, the most important part is the clock generation. The testbench writes into the clock control register and selects the clock required for initialization and then changes the clock frequency for data transfer. The output from the physical view verification should be the same as the logical view.

5.3 Algorithm in the C testbench

The algorithm for the c testbench remains the same in both the logical view and the physical view.

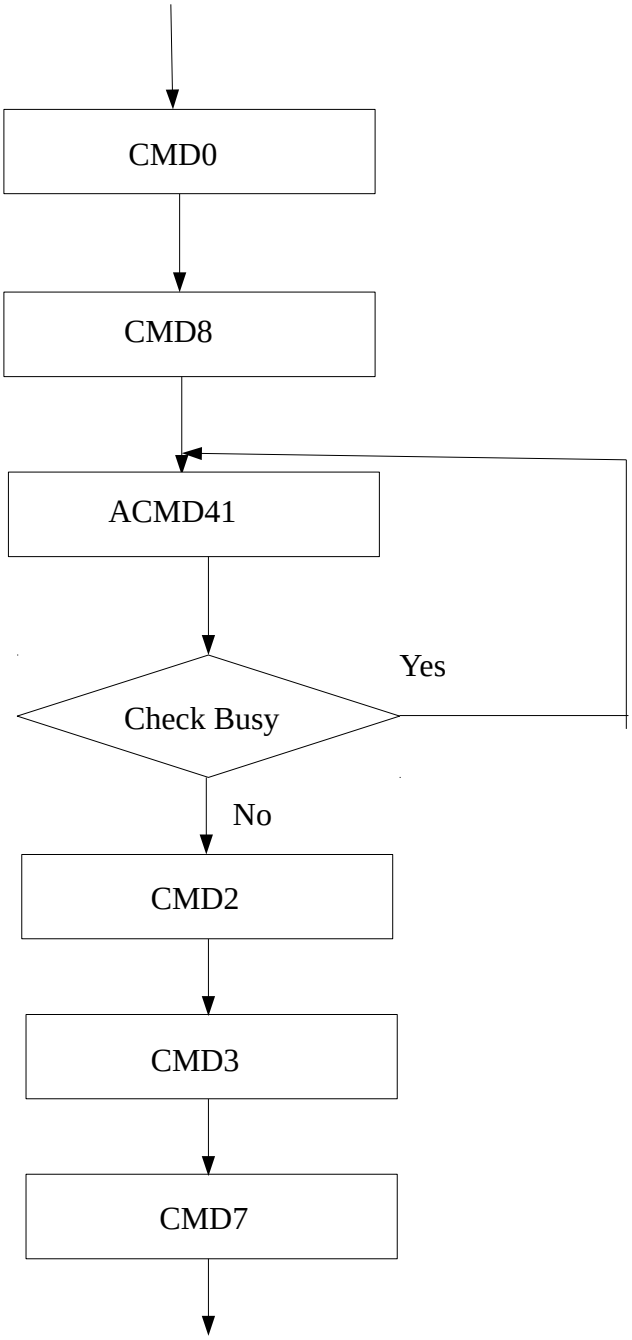

Figure 5.1 : Algorithm for Initialization

After this algorithm is finished successfully the driver can send any of the read or write commands for data read or write. CMD17 and CMD18 are sent for single and multiple block reads respectively. CMD 24 and CMD25 are used for single and multiple block writes respectively.

6. Results

The logical view gives the successful initialization and successful data sent and received for both single and multiple block read and write.

In the physical view, it is seen that the behavior of the host controller is similar to that of the logical view. The timing is noted here.

To send a command at 400Khz we need 120µsec as expected.

To send a command at 25 Mhz we need 1.92µsec.

To process a single block read, it takes close to 4µsec of time.

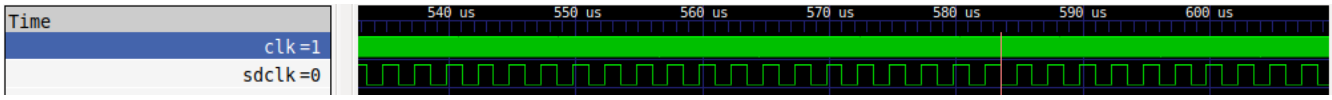

Figure 6.1 : 400 KHZ SD clock generated from clk

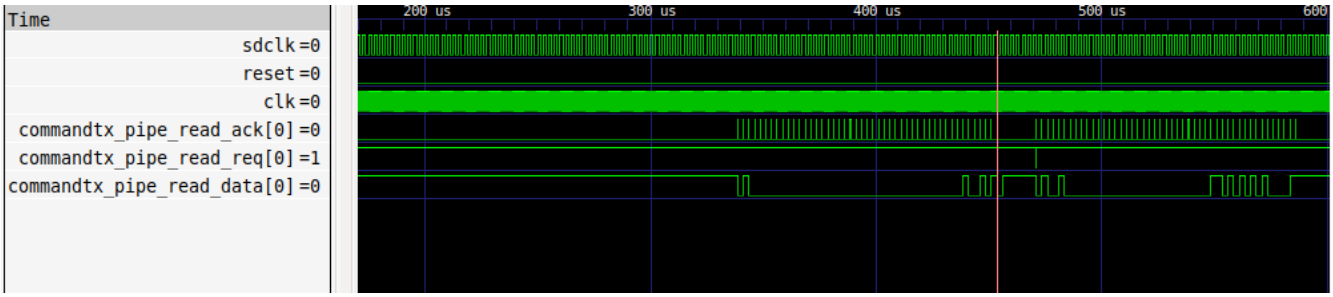

Figure 6.2 : Command Transmission at 400 KHZ

It is seen in the Fig 6.2 that the time taken in between two succesive commands in 15µsec.

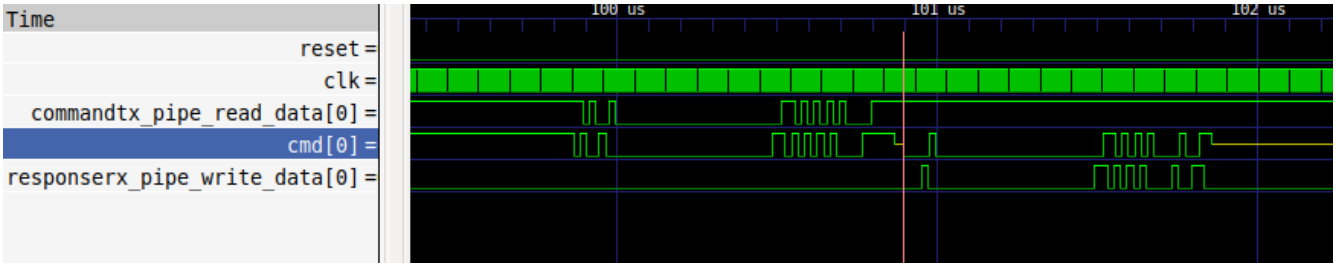

Figure 6.3 : Two succesive commands sent

In Fig 6.3 cmd is the tristate line sending the data to the sd card and commandtx_pipe_read_data and the responserx_pipe_write_data are the FIFOs for the command and response respectively.

Here in the simulation, the time gap between the end bit of command and the first bit of response is less than which is less than the best case. The response is sent by the sd card in between 1 sd clock cycle to 8 clock cycle.

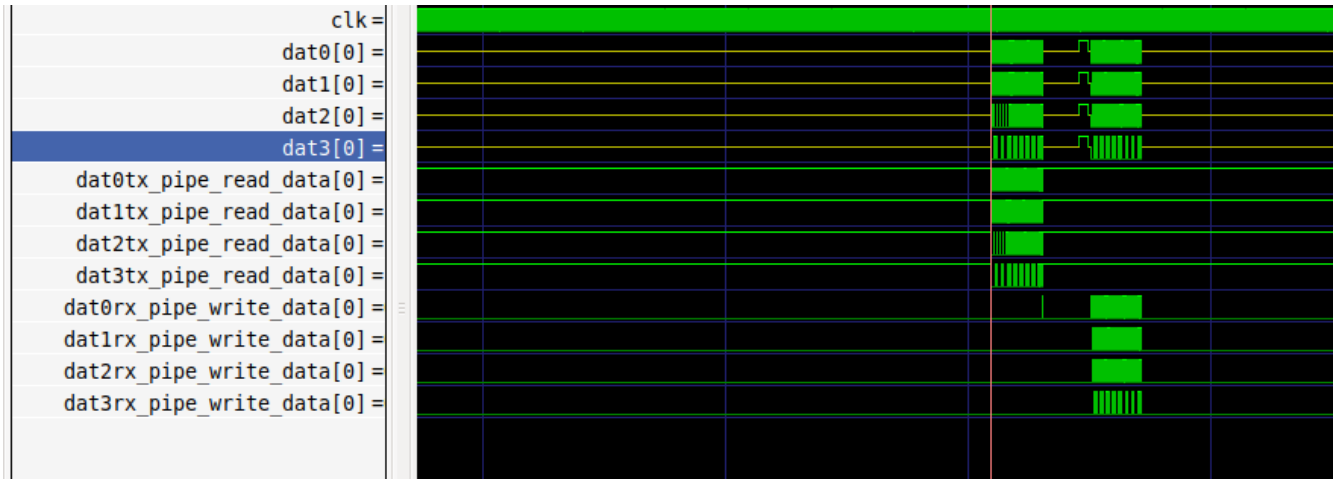

Figure 6.4 : Data sent and received trhough the data lines

In Fig 6.4 dat[0] to dat[3] are the tristated data lines and dat0tx to dat3tx are the FIFOs used for data transmission similarly dat0rx to dat3rx are the data receiver FIFOs.

Here it can be seen that the data is broken into 4 lines for both transmission and reception and all of them work simultaneously to maximize the performance of the sd host controller.

Also, it can be seen that the data has been separated from the tristate to the FIFOs successfully.

7. Future Work

The design needs to be validated in an FPGA with a real SD card. The FPGA of choice can be Basys3 board by digilent. The SD card can be connected through the PMOD breakout board for SD card. The design needs to be synethesized and the hardware needs to fit in the board. The Basys3 board has 20800 LUTs as logic. So the design should not produce more than this amount of hardware. The same testbench used in physical view verification can be used here also. The board will connect with the terminal using UART communication.

A. Abbreviations Used

Abbreviation	Full Form
SD	Secure Digital
FPGA	Field Programmable Gate Array

References

[1] Generic AJIT processor core description and user guide by Prof. Madhav P. Desai

[2] SD Specifications Part 1 PHYSICAL LAYER Simplified Specification Version 3.01 by SD Assosciation.

[3] SD Specifications Part A2 SD Host Controller Simplified Specification Version 3.00 by SD Assosciation

[4] SD card Wikipidia

[5] How to Use MMC/SDC – elm-chan.org/docs/mmc/