A Report

On

Designing and Simulation of Serial Communication Protocol

By

Name of the student	Registration number
BALAJI RAO VAVINTAPARTHI	1700298C204

Prepared in the partial fulfilment of the

Practice School II Course

At

MOSCHIP Technologies Pvt. Ltd.

Plot No 83 & 84, level 2, Punnaiah Plaza,

Road No 2, Banjara Hills, 500034

Hyderabad

A Practice School II Station of

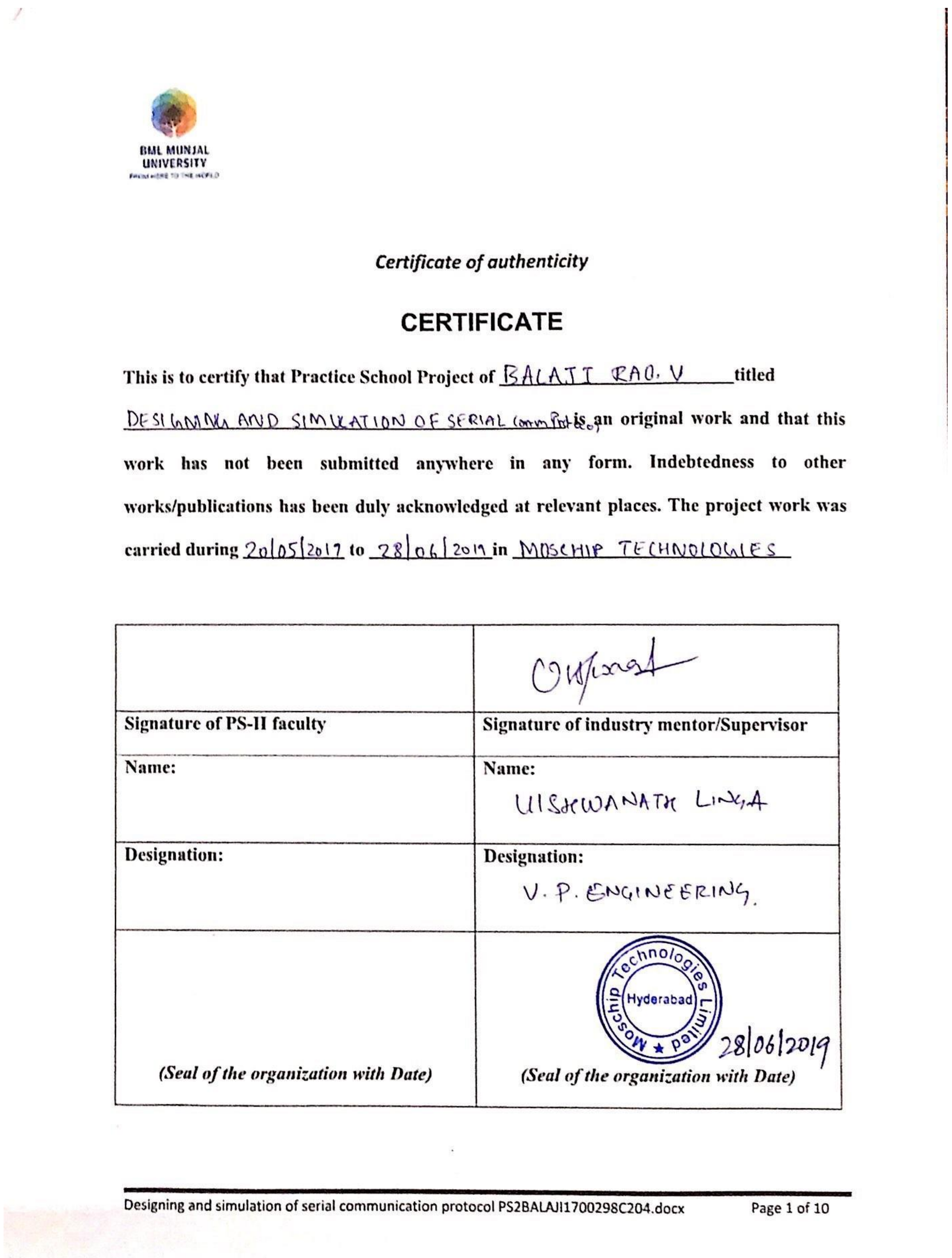
BML MUNJAL UNIVERSITY

July 2019

BML MUNJAL UNIVERSITY
FROM HERE TO THE WORLD

Table of Contents

Completion Certificate



Certificate of authenticity

CERTIFICATE

This is to certify that Practice School Project of BALAJI RAO. V titled DESIGNING AND SIMULATION OF SERIAL Comm Protocol is an original work and that this work has not been submitted anywhere in any form. Indebtedness to other works/publications has been duly acknowledged at relevant places. The project work was carried during 20/05/2017 to 28/06/2019 in MOSCHIP TECHNOLOGIES

	[illegible]
Signature of PS-II faculty	**Signature of industry mentor/Supervisor**
Name:	**Name:** VISHWANATH LINGA
Designation:	**Designation:** V. P. ENGINEERING.
(Seal of the organization with Date)	Moschip Technologies Limited Hyderabad 28/06/2019 *(Seal of the organization with Date)*

Designing and simulation of serial communication protocol PS2BALAJI1700298C204.docx Page 1 of 10

Acknowledgements

The internship of 6 weeks I had at MOSCHIP Technologies Pvt. Ltd. was a great opportunity to learn professional environment and work. I would like to thank the Dean of SoET, Dr. M.B. Srinivas for providing the students with this opportunity.

Further extending my acknowledgement to the Co-ordinator of the program Dr. Maheshwar Dwivedy and Mr. Kosthuba Nand for making all the formalities clear.

I sincerely thank my faculty mentor Dr. M.B. Srinivas who always kept me updated and made my internship a healthy one. I would also like to thank my industry mentor Mr. Umesh Singh for helping me with any technical issues in the project and guiding me whenever and wherever required.

Objectives

Designing and simulation of a serial communication protocol by using hardware description language, Verilog. The main objective of this project is to understand the principle of communication protocol and their implementation in the field of VLSI (Very Large-Scale Integration).

Problem Statement

To simulate a serial communication protocol using HDL (Hardware Description Language) Verilog and implement it on a Xilinx FPGA (Field Programmable Gate Array) board.
The main serial communication protocol focused in this project is I^2C (Inter Integrated Circuit) protocol. This protocol is basically used to send data to peripheral devices.

So, the working of this protocol which includes data transmission and the communication between master and slave should be understood.

In order to simulate this protocol first the basics of Verilog must be known. This HDL is very important and useful in VLSI. This HDL will allow fast design and better verification.

In most of the industries Verilog is used for designing all types of circuits. The tool used to perform this design and simulation is Xilinx Vivado.

Company Profile

Moschip Technologies Limited, formerly Moschip Semiconductor Technology Limited, is a fabless semiconductor company. Established in 1999, MosChip is the First Fabless Semiconductor company publicly traded in India with approx.20 years of experience. The Company's principal activity includes software development and designing. The Company is engaged in the business of development and manufacturing of System on Chip (SOC) technologies. The Company focuses on providing value added services in very-large-scale integration (VLSI) design, application specific integrated circuits (ASICs), software development and development system on a chip (SOC) for aerospace and defence, consumer and industrial applications and Internet of Things/Everything (IOT/E) products and services across various industries.

Over the past 2 decades, MosChip has developed and shipped millions of ASIC connectivity ICs. MosChip's IoT solutions like Smart Lighting, Smart Metering and Asset Tracking are successfully deployed in volume.

Website:

http://www.moschip.com

Headquarters:

Hyderabad, Andhra Pradesh

Year Founded:

1999

Company Type:

Public Company

Specialties:

SOC, ASIC, FPGA, PCB, Design, Verification, Synthesis, IoT Services, IoT, Systems, Testing, remote monitoring, smart lighting, smart street lighting, M2M, IoT Analytics, RTL Design, Design Verification, Heavy Equipment Monitoring, Semiconductor IPs, Mixed Signal IPs, Verification IPs, Design IPs, Turnkey ASIC solution, mobile and enterprise applications, embedded systems, system engineering, VLSI training.

Methodology

Before getting started, the basics and importance of communication protocols and Hardware Description Language (HDL) should be known.

So, what is a communication protocol? And how is it important?

i. Communication protocols:

In the field of telecommunication, a communication protocol is a system of rules that allow two or more entities of a communication system to transmit information via any kind of variation of a physical quantity.

These protocols can be implemented by hardware, software or combination of both.

ii. Types of communication protocols:

Basically, data can be transmitted between a sender and receiver in two main ways

- **Serial communication**
- **Parallel communication**

Serial communication:

It is the method of transferring one bit of data at a time through a medium.

There are different types of serial communication protocols which include

1. I^{2}C protocol (Inter Integrated Circuit)
2. SPI protocol (Serial Peripheral Interface)
3. UART protocol (Universal Asynchronous Receiver-Transmitter)
   etc.,

Parallel communication:

It is the method of transferring blocks of data at the same time over multiple mediums.

There are different types of parallel communication protocols which include

1. ISA protocol (Industry Standard Architecture)
2. ATA protocol (Advance Technology Attachment)
3. SCSI protocol (Small Computer System Interface)
   etc.,

In this project only I^{2}C (Inter Integrated Circuit) protocol which is a serial communication protocol is focused.

iii. I^{2}C (Inter Integrated Circuit) Protocol:

I^{2}C protocol is a serial communication protocol and is used for two wire interfaces to connect low speed devices and peripheral systems. The two wires in this bus protocol are SDA (Serial Data) line and SCL (Serial Clock) line. I^{2}C is synchronous so the output of bits is synchronized to the sampling of the bits by a clock signal shared between the master and slave.

This protocol is not restricted to single master and single slave it can be a multi master multi slave interface. It is also bidirectional, so data can be transferred or received through the data line. It is used for short distance, inter board communications.

SDA (Serial Data) line- It is used by the master and slave to send and receive data.

SCL (Serial clock) line- It carries the clock signal.

The standard mode speed is up to 100 kbps but there are various modes which go up to 5 Mbps. The max no of masters which can be used here is unlimited, but the max no of slaves depends on the no bits in the address, if it is a 7-bit address then the max no of slaves is 127.

So, in I^{2}C data is transferred in messages. Messages are broken into frames of data. Each message contains:

1. START condition
2. 7-bits address frame
3. ACK (acknowledgment) bit for the address
4. 8-bits data frame
5. ACK (acknowledgment) bit for the data
6. STOP condition.

By this way the data can be transferred or received using this communication protocol.

1. START condition:
   The SDA line switches from high voltage level to a low voltage level before SCL line switches form high to low.

2. Address frame:
   A 7- bit sequence unique to each slave that identifies the slave when the master wants to send or receive data from it.
   A single bit specifying whether the master is sending data to the slave (low voltage level) or requesting data from it (high voltage level) is called a READ/WRITE bit. It is generally the last bit of the address frame.

3. ACK/NACK bit:
   Each frame in a message is followed by an acknowledge or non-acknowledge bit. If an address frame or data frame was successfully received, an ACK bit is returned to the receiving device.

4. Data frame:
   Data frame is nothing but the data the master wants to send or receive from the slave. It is generally an 8- bit frame. So, the data sent is broken to 8-bits frame and then transferred.

5. STOP condition:
   The SDA line switches from low voltage level to a high voltage level after SCL line switches form low to high.

The communication between a master and a slave begins with a START condition followed by the slave address (which is 7 bit) to be reached, one read/write bit a bit of recognition that can be ACK (acknowledgment) if the communication was successful and NACK (negative acknowledgment) if the communication was unsuccessful, the 8 bits of data to send or to receive, the ACK or NACK bit and finishes with a STOP condition or a condition repeated START.

So, this is how I^{2}C communication protocol works. Based on these conditions a program is designed and simulated. After the simulation is completed a waveform is obtained which is the output. The whole program is written using the Hardware Description Language (HDL) Verilog and the tool used to perform this operation is Xilinx Vivado.

(**Verilog** is a textual format for describing the electronic circuits and systems. It is used for verification through simulation, for timing, test analysis and logic synthesis. It is a very important language in the field of VLSI.)

iv. I^{2}C protocol program:

a) Program file:

```
timescale 1ns / 1ps
module project_2(
input wire clk,
input wire reset,
output reg i2c_sda,
output wire i2c_scl,
output reg [7:0] out_data
);

localparam state_idle = 0;
localparam state_start = 1;
localparam state_addr = 2;
localparam state_rw =3;
localparam state_ack=4;
localparam state_data=5;
localparam state_ack2=6;
localparam state_stop=7;

reg [7:0] state;
reg [6:0] addr;
reg [7:0] count;
reg [7:0] data;
reg i2c_scl_2=0;

assign i2c_scl = (i2c_scl_2 == 0) ? 1 : ~clk;
```

```
always@(negedge clk) begin

if (reset==1) begin

i2c_scl_2<=0;

end


else begin

if ((state==state_idle) || (state==state_start) || (state==state_stop)) begin

i2c_scl_2<=0;

end

else begin

i2c_scl_2<=1;

end

end

end


always@(posedge clk) begin

if (reset==1) begin

state<=0;

i2c_sda <=1;

addr <=7'h50;

count <=8'd0;

data <=8'haa;

end


else begin

case(state)
```

```
state_idle :begin
i2c_sda<=1;
state <= state_start;
end

state_start : begin
i2c_sda <=0;
state <= state_addr;
count <=6;
end

state_addr: begin
i2c_sda<= addr[count];
if (count==0) state<= state_rw;
else count <= count-1;
end

state_rw: begin
i2c_sda <=1;
state <= state_ack;
end

state_ack: begin
i2c_sda<=0;
state <= state_data;
count <= 7;
end
```

```
state_data: begin
i2c_sda<= data[count];
if (count==0) state<= state_ack2;
else count<= count-1;
end

state_ack2: begin
state<= state_stop;
end

state_stop: begin
i2c_sda <=1;
state<= state_idle;
end
endcase
end
end

always @(negedge clk) begin
out_data=data;
end

endmodule
```

In the above program there are different states as described in the methodology. Different wires and registers are declared as inputs and outputs. Wires only carry the data they do not store the values. But registers are used to store the defined values.

Different states are used starting from state idle to state stop. Counters are used to count the values for reading the address and the data.

Here the address and data are predefined. The address is a 7-bit hexadecimal number 50 and the data is an 8-bit hexadecimal number aa. The address and data can have any value, but the address should only be a 7-bit number and data should only be a 8-bit number.

So now this program needs a test bench file. The design is instantiated in a test bench, stimulus is applied to the inputs and the outputs are monitored for the desired results.

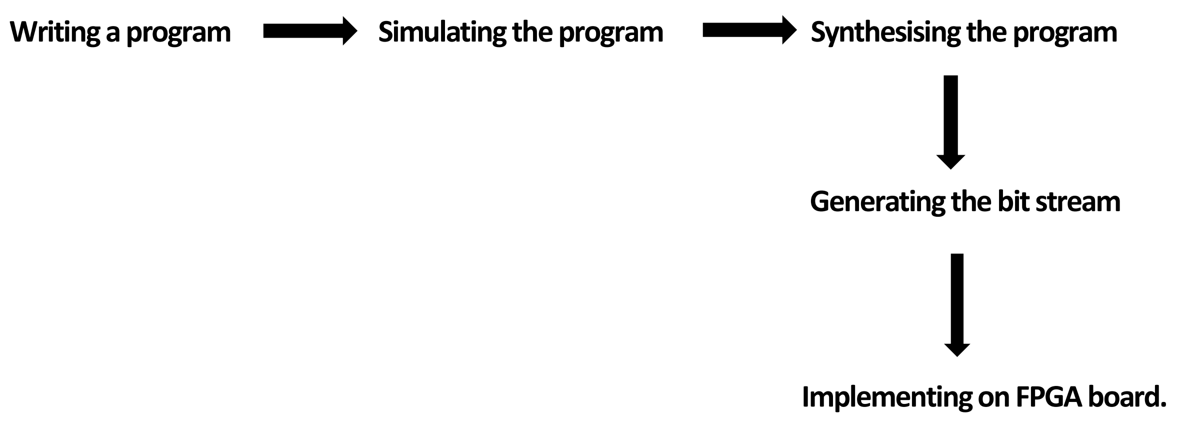
b) Test bench file:

```
module test_bench;
reg clk;
reg reset;
wire i2c_sda;
wire i2c_scl;

 project_2 uut (
 .clk(clk),
 .reset(reset),
 .i2c_sda(i2c_sda),
 .i2c_scl(i2c_scl)
 );

initial begin
clk=0;
forever begin
```

```
#5 clk= ~clk;
end
end

initial begin
reset = 1;
#10;
reset=0;
#100;
$finish;
end
endmodule
```

v. Implementation:

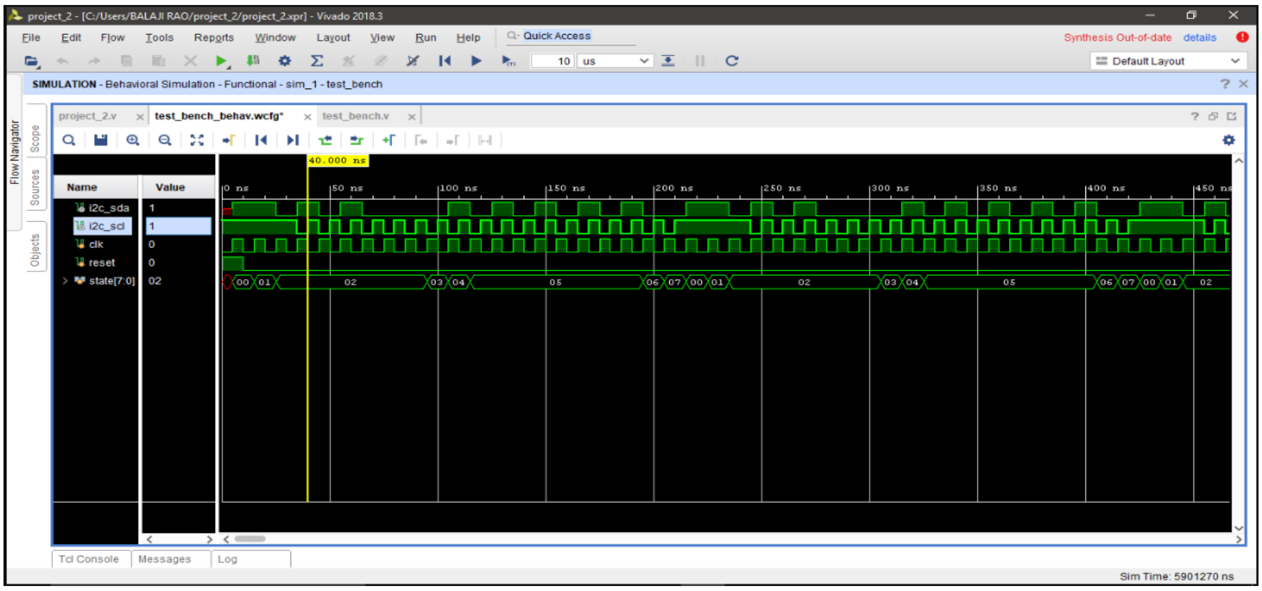
The written program will be implemented on a Xilinx FPGA (Field Programmable Gate Array) board. But to implement on a FPGA board firstly the FPGA flow should be followed which includes:

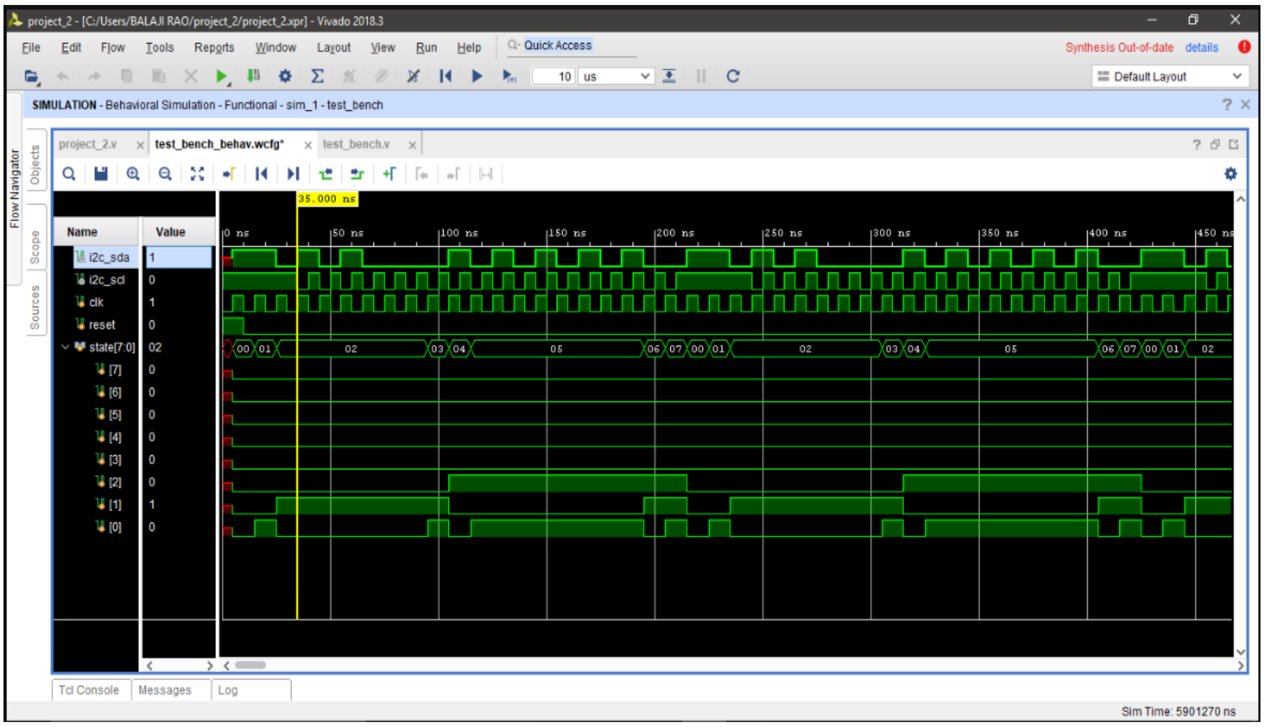


Results and Discussion

Once the program is designed then it is simulated. Simulation is used to verify the functional correctness of a digital design that is modelled using our HDL. This simulation process gives a waveform which is the output.

The simulated result of the above program is:





In the output waveform the values can be observed at different states.

At

State 0: it is the ideal state where both the SDA and the SCL are high.

State 1: in this state the SDA line drops to low before the SCL line drops to low, which is the START condition.

State 2: In this state the 7-bit address frame can be observed including the 3rd state bit.

State 3: In this state the R/W bit can be observed which is the last bit of the address frame.

State 4: In this state the ACK bit for the address can be observed.

State 5: In this state the 8-bit data frame can be observed.

State 6: in this state the ACK bit for data can be observed.

State 7: In this state the SDA line goes to high after the SCL line goes to high, which is the STOP condition.

Once when the simulation and synthesis were completed this program was implemented on a Xilinx FPGA board. As it was a defence project board, pictures related to the board and the synthesized output results were not allowed to use in the report.

The above result is the simulated output of the program.

Conclusions

Communication protocols have various uses in telecommunication. Only serial communication protocol (I^2C) was focussed in this project but using the hardware description language other communication protocols can also be implemented on FPGA.
Different circuits can be designed using Verilog as it is a high-level programming language that runs concurrently, to the difference with other programming languages which works sequentially, so it performs a faster and more efficient communication when implementing the communication protocol.
Concluding in the field of VLSI using Hardware Description Language (HDL) and Field Programmable Gate Array (FPGA) many circuits can be designed and synthesized.

References

1. ***Verilog HDL: A Guide to Digital Design and Synthesis***, *Second Edition By Samir Palnitkar.*
2. ***Hardware modelling using Verilog –*** *I Sengupta*
3. *Understanding I2C bus by- Texas Instruments:* *https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=2ahUKEwiTj_HDpuPiAhXQ6nMBHenUBDsQFjAAegQIABAC&url=http%3A%2F%2Fwww.ti.com%2Flit%2Fan%2Fslva704%2Fslva704.pdf&usg=AOvVaw0MO8lWY9iHs3PZqsN6fgWm*
4. *Basics of I2C communication protocol- Circuit basics:* *https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=6&cad=rja&uact=8&ved=2ahUKEwiTj_HDpuPiAhXQ6nMBHenUBDsQFjAFegQIBhAB&url=http%3A%2F%2Fwww.circuitbasics.com%2Fbasics-of-the-i2c-communication-protocol%2F&usg=AOvVaw1CaSZeLrDMATHGcj5u_SCP*
5. *I2C protocol -Wikipedia:* *https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=28&cad=rja&uact=8&ved=2ahUKEwihlOvMwK_jAhUBN48KHVWKB_4QFjAbegQIBBAB&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FI%25C2%25B2C&usg=AOvVaw3Z6tnLdIOpEwbONryMwtAd*