I2C BUS PROTOCOL USING VERILOG

Prinkle Talan, Anshul Tyagi,
Assistant Professor, Student,
Electronics & Communication Engineering,
Maharaja Surajmal Institute of Technology, Delhi, India

Abstract : This research paper is about the designing of I2C communication protocol single master using Verilog language. This is a bidirectional, two wired serial bus which is used to transport the data between integrated circuits. Slave devices are interfaced to the microcontroller with the help of the I2C bus. The protocol concept comes into the picture for reducing the hardware complexity and power consumption. The I2C protocol used to connect a maximum of 128 devices that are all connected to communicate with the SCL and SDL lines of the master unit as well as the slave devices. I2C Module and its test bench is simulated in modelsim as well as in Xilinx ISE.

IndexTerms **- I2CBusProtocol, SDA, SCL, Verilog, ModelSim.**

I. INTRODUCTION

The I2C stands for **"Inter Integrated Circuit"**. A I2C bus is a bidirectional two-wired serial bus which is used to transport the data between integrated circuits as shown in **Fig.1**. It was first introduced by the **Philips semiconductors in 1982.**

In Philips Semiconductor, **"I2C Bus Specification", version 2.1 January 2000,** includes the all detailed functionality and specifications along with the minor modification like after a repeated START condition in Hs-mode (high speed), it is possible to stretch the clock signal and Some timing parameters in Hs-mode have been relaxed.[1] The updated version **Philips Semiconductor, "I2C manual", March 2003,** explained a broad overview of the various serial buses, why the I2 C bus should be considered, technical detail of the I2 C bus and how it works, previous limitations/solutions.[2] Also **Philips Semiconductor , "I2C-bus specification and user manual", April 2014** explains new ULTRA-Fast Mode specification of I2C along with all updated. The Ultra Fast-mode is a uni-directional mode with data transfers of up to 5 Mbit/s. Serial, 8-bit oriented, bidirectional data transfers can be made at up to 100 kbit/s in the Standard-mode, up to 400 kbit/s in the Fast-mode, up to 1 Mbit/s in the Fast-mode Plus (Fm+), or up to 3.4 Mbit/s in the High-speed mode. [3]

Nowadays the protocols play an essential role in the **embedded system design.** Without going to the protocols, if we want to expand the peripheral features of the microcontroller, the complexity and power consumption will increase. There are different types of bus protocols available such as **USART, SPI, CAN, I2C bus protocol**, etc., which are used for transferring the data between two systems. Transmitting and receiving the information between two or more than two devices require a communication path called as a bus system.

The I2C bus consists of three data transfer speeds such as **standard, fast-mode and high-speed-mode.** The I2C bus supports **7-bit and 10-bit address** space device and its operation differ with low voltages. It is only intended for short distance communications.[4][5]

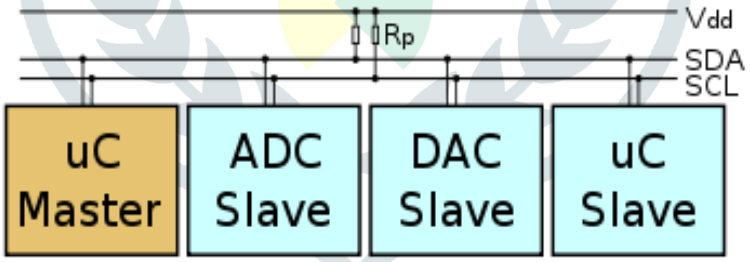

Fig.1 Block diagram of connecting devices using I2C

II. FEATURES

- **SDA (Serial Data)-** The line for the master and slave to send and receive data.
- **SCL (Serial Clock)-** The line that carries the clock signal.
- I2C is a serial communication protocol, so data is transferred bit by bit along a single wire **(the SDA line)**. Both need to be pulled up with a resistor to **+Vdd**. Like SPI, I2C is synchronous, so the output of bits is synchronized to the sampling of bits by a clock signal shared between the master and the slave. The clock signal is always controlled by the master. With I2C, data is transferred in **messages** as shown in **Fig.2.** Messages are broken up into ***frames*** of data. Each message has an address frame that contains the binary address of the slave, and one or more data frames that contain the data being transmitted. The message also includes start and stop conditions, read/write bits, and **ACK/NACK** bits between each data frame[1][2][3]
- **Start Condition-** The SDA line switches from a high voltage level to a low voltage level *before* the SCL line switches from high to low.
- **Stop Condition-** The SDA line switches from a low voltage level to a high voltage level *after* the SCL line switches from low to high.
- **Address Frame-** A 7 or 10 bit sequence unique to each slave that identifies the slave when the master wants to talk to it.
- **Read/Write Bit:** A single bit specifying whether the master is sending data to the slave (low voltage level) or requesting data from it (high voltage level).
- **ACK/NACK Bit-** Each frame in a message is followed by an acknowledge/no-acknowledge bit. If an address frame or data frame was successfully received, an ACK bit is returned to the sender from the receiving device.

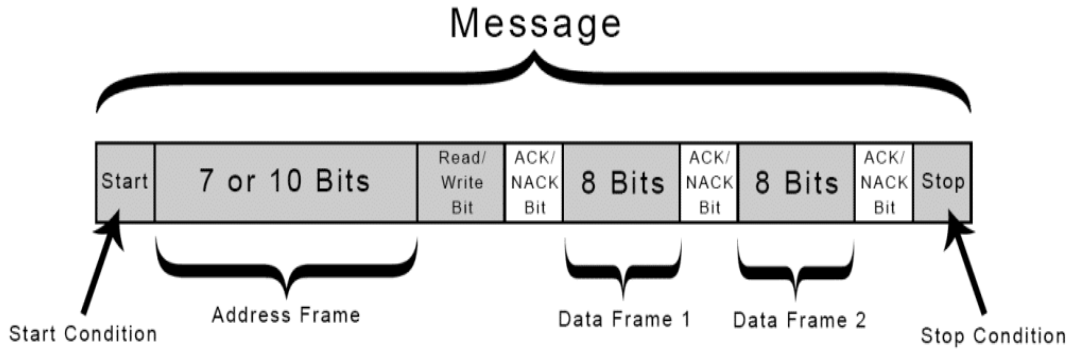

Fig.2. Message Format in I2C

III. I2C PROTOCOL

1. The master sends the start condition to every connected slave by switching the **SDA line** from a high voltage level to a low voltage level *before* switching the **SCL line** from high to low.
2. The master sends each slave the **7 or 10 bit address** of the slave it wants to communicate with, along with the **read/write** bit.
3. Each slave compares the address sent from the master to its own address. If the address matches, the slave returns an **ACK** bit by pulling the SDA line low for one bit. If the address from the master does not match the slave's own address, the slave leaves the **SDA line** high.
4. The master sends or receives the data frame.
5. After each data frame has been transferred, the receiving device returns another ACK bit to the sender to acknowledge successful receipt of the frame.
6. To stop the data transmission, the master sends a stop condition to the slave by switching SCL high before switching SDA high.

The **Fig.3.** shows the I2C bus waveform during transmission operation. Initially both the lines are high and when start condition occurs SDA only goes high to low and then transmits address and data serially. How does it transmit address and data serially is explained in next sections. At stop condition SDA goes low to high only when SCL is high.[1][10]

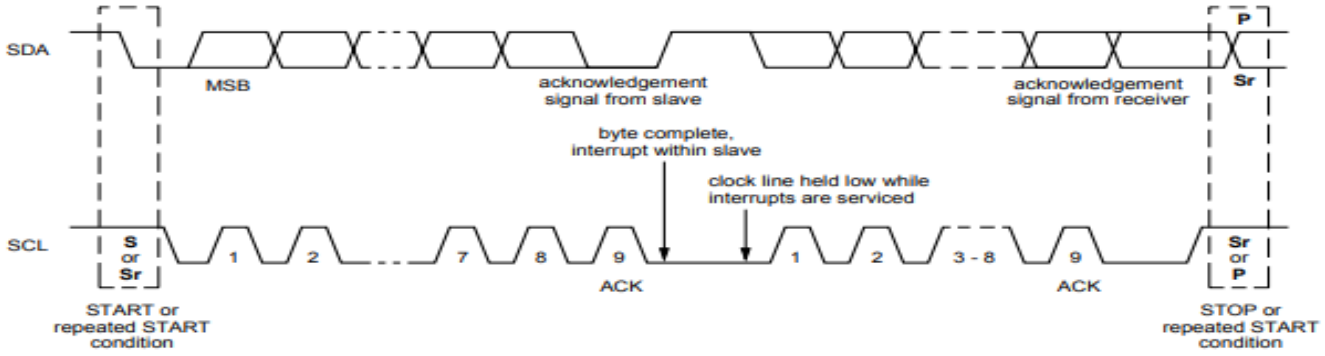

Fig.3. I2C Protocol Waveforms

IV. DESIGN ALGORITHIM

The FSM (Finite State Machine) for I2C is designed according to its communication protocol, Features and functionality. Various steps of communication are represented by the states of FSM as shown in **Fig. 4.**

Following are the stages of FSM of I2C:-

- **IDLE:** At this stage no action will be performed by the bus. And enable is also low.
- **START:** when enable signal is switch to high it goes into start stage and start condition are applied which explained earlier. If enable is found to be '0' it will go back to IDLE stage.
- **ADDR:** At this stage the address of the slave is transmitted serially by using the **count register**. It will remain on this stage until count is not equal to '0'.
- **R/W:** When the count is '0'i.e address has been sent then it checks for Read/Write signal. If It is 1 Read operation is to be performed otherwise Write Operation will take place.
- **ACK:** This stage transfer the control to the DATA TRANS stage if acknowledgement signal is 1 received from slave it goes to DATA TRANS stage otherwise it will go to STOP condition.
- **DATA TRANS:** In this stage Data is transmitted to the slave serially by using the count register.
- **STOP:** This stage occur when ACK=0 and stop condition are applied as explained earlier. If enable signal is found to be high then START stage occurs otherwise it will go into IDLE STAGE.[5][10]

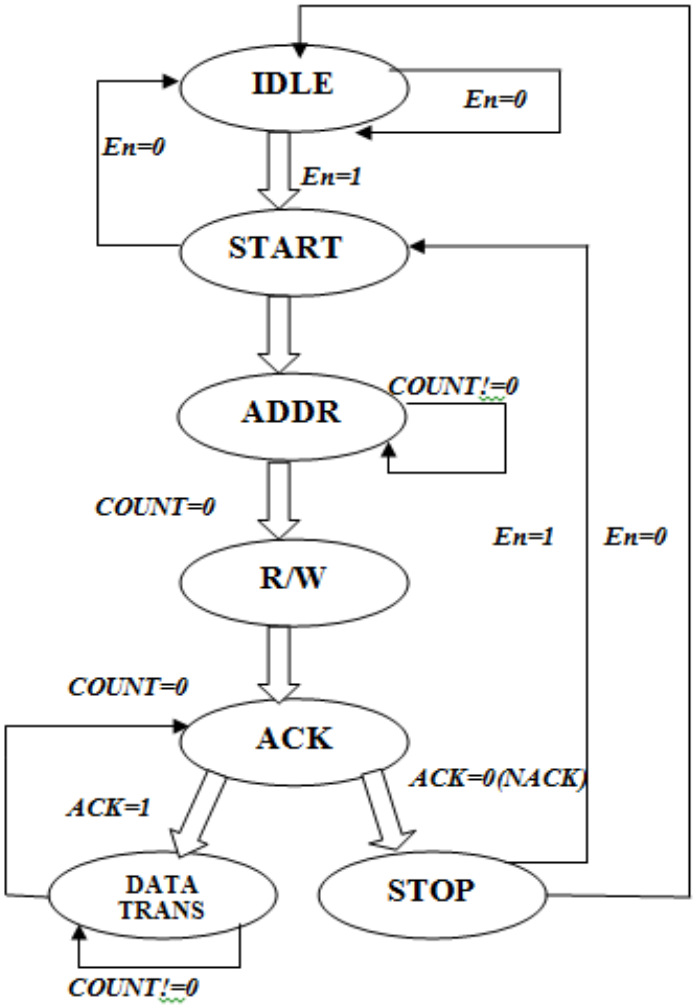

Fig.4. FSM Design for I2C Bus

V. VERILOG CODE METHODOLOGY

At last this FSM for I2C bus master will be converted into Verilog code and this code will be simulated using Verilog simulator. Here the Verilog code is simulated by using two simulator **MODELSIM (PE Student Edition 10.4a) & ISE XILINX (ISE Design Suite 14.6)**.

The simulated output from both the simulators has been shown and explained in Simulation Section. The various stages are of I2C become the various stages of CASE(stage) . **While coding one thing should be kept in mind that data bit on SDA line will change only when SCL line is going LOW.**

A Variable **Count** type reg can be declared to transmit the data serially. **So from ADDR stage to NACK should be performed only when SCL ==0.** CASE should be in always block which is active only at the positive edge of the CLOCK (CLK) signal. Switching from one case stage to another is take place according to the FSM of I2C.[6][7]

VI. SIMULATION

This section contains the Output waveform for I2C bus Protocol which is obtained from I2C Verilog code designed in Verilog simulator using the FSM (shown earlier) **MODELSIM (PE Student Edition 10.4a**) from Mentor Graphics **& ISE XILINX (ISE Design Suite 14.6)** from Xilinx has been used.

Simulation is the process of using a simulation software (simulator) to verify the functional correctness of a digital design that is modeled using a HDL (hardware description language) like Verilog. Explanation of Simulation is given below:-

Signal in waveforms are:

- SCL(Serial Clock line),
- SDA(Serial Data line),
- EN(Enable),
- CLK.

Each action is takes place at the positive edge of the clock signal means whenever the clock transition is take place from 0 to 1 different stages from **FSM** is followed.

The **EN (Enable)** signal is kept low for 10 ns initially then after it is active and further functions takes place. The first simulated output waveforms is obtained from the **Modelsim** simulator shown in **Fig.5.** and the second one is obtained from **Xilinx** simulator shown in **Fig.6**. Both are showing the same waveforms for I2C protocol. As we can see from simulated output that for very first clock (0 to10 ns) the enable pin is low and bus master will go into IDLE stage hence nothing is performed by I2C so that SCL and SDA both are high. At 10 ns second the enable pin is high and master into START stage and following the start condition.

At next positive edge master goes into ADDR stage and address of the slave is transmitted serially and when count become 0 it will go into R/W stage and check for read/write single and got ACK from slave and then go into the DATA TRANS stage and transmitted the data serially. And then goes into stop stage and the SCL and SDA again high. If NACK signal is received from slave

then it will go into STOP stage. From output waveform it is clear that IDLE, START and STOP stage SCL and SDA are high only.[9][10]

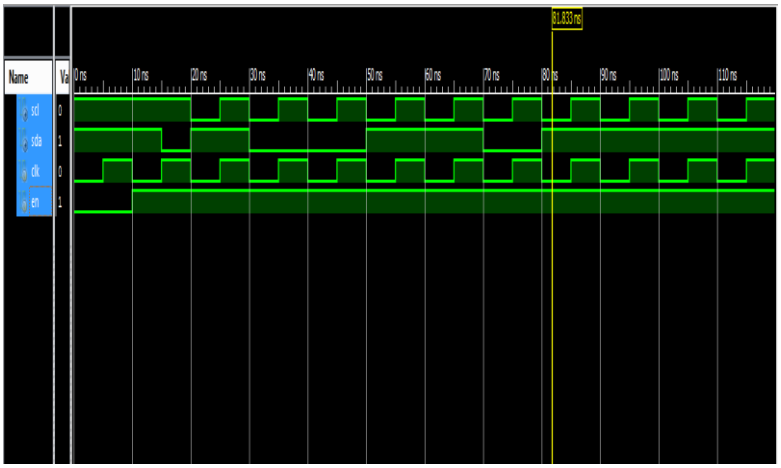

Fig.5. Output Waveforms from Xilinx

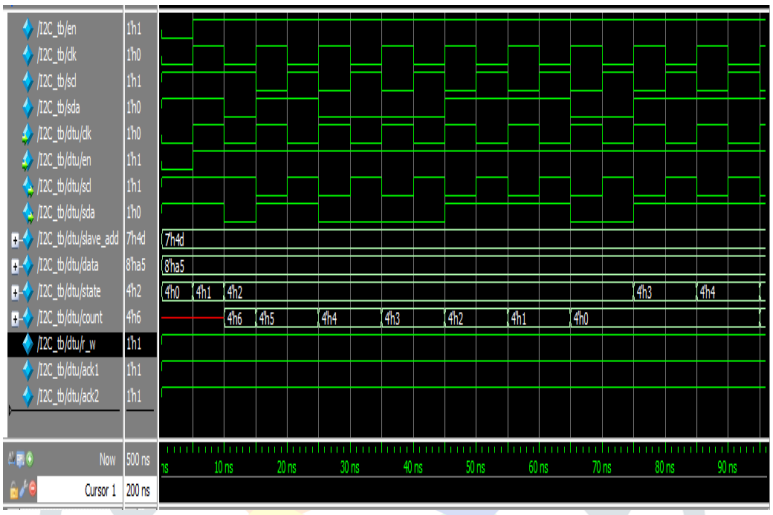

Fig.6. Output Waveforms from Modelsim

VII. SYNTHESIS

This section contains the RTL diagram of I2C Bus master which is obtained after completed the synthesis of I2C Verilog code successfully on Xilinx ISE tool. Modelsim cannot perform Synthesize operation. **ISE XILINX (ISE Design Suite 14.6)** is used for synthesis.

Synthesis is a process in which a design behavior that is modeled using a HDL is translated into an implementation consisting of logic gates. This is done by a synthesis tool which is another software program.

For synthesis, the design behavior should be modeled at an **RTL (Register Transfer Level**) abstraction. This means modelling in terms of the flow of digital signals between hardware registers (flip-flops) and the logical operations (combinational logic) performed on those signals. Hence if a Verilog design model is intended for synthesis, then only synthesizable constructs should be used, while for simulation there are no such restrictions. Yes it may happen that the same code will not give successful synthesis which gives complete simulated output waveforms. This problem occurs due to use of non-synthesizable constructs in Verilog. Also this may occur due to **mixing of Blocking & Non Blocking Assignments or multiple drivers of a signal**. These things can be simulated properly but can not be synthesized.

Fig.7. shows the RTL ***(Register Transfer Level)*** which can be obtained only after the successful synthesizable code. After it Implementation and Generation Program file can be done to implement the design on FPGA Board.[8]

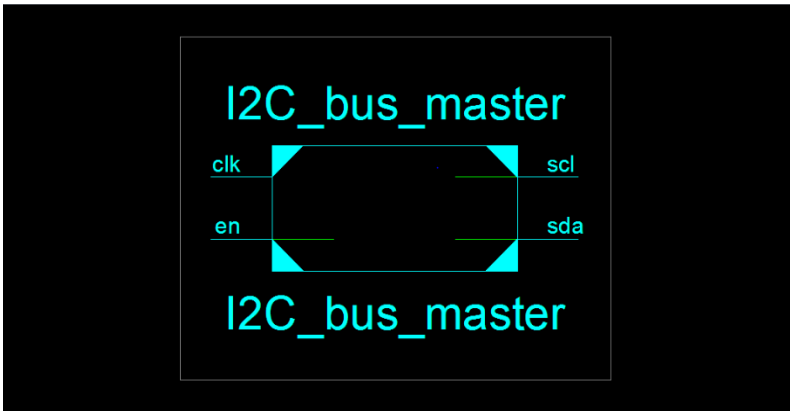

Fig.7. RTL diagram of I2C Bus Master

VIII. CHALLANGING PHASE

The problem during coding is that **START and STOP** condition are quite difficult to achieve. There can be various ways to achieve these conditions but the way which is found be suitable for appropriate waveforms of I2C protocol is given here.

This problem can be solved by putting the condition **if(SCL==0)** in begin of each stage except **IDLE, START and STOP** stages.so that whenever data changes from 1 to 0 or 0 to 1 the SCL is always '0'

IX. RESULT

Data is successfully READ and WRITE by the master and also data is transmitted serially.

X. COMPARATIVE STUDY

There is a lot of serial communication protocol like UART, I2C and SPI etc. but I2C and SPI and are very famous among them ,I2C made by the Philips (Nowadays NXP) and SPI is made by the Motorola. Both protocols are commonly used in electronic devices.

There are few points which can compare them easily.

A. *UART*

- It is simple communication and most popular which is available due to UART support in almost all the devices with 9 pin connector. It is also referred as RS232 interface.
- They are suitable for communication between only two devices.
- It supports fixed data rate agreed upon between devices initially before communication otherwise data will be garbled.

B. *SPI*

- It is simple protocol and hence so not require processing overheads.
- Supports full duplex communication.
- SPI uses less power compare to I2C .
- To add a device in SPI requires one to add extra CS line and changes in software for particular device addressing is concerned.
- SPI can be multi-save but does not a multi-master serial protocol, whereas I2C can be multi-master and multi-slave,
- No flow control available in SPI.

C. *I2C*

- Due to open collector design, limited slew rates can be achieved.
- More than one masters can be used in the electronic circuit design.
- Needs fewer i.e. only 2 wires for communication.
- I2C addressing is simple which does not require any CS lines used in SPI and it is easy to add extra devices on the bus.
- It uses open collector bus concept. Hence there is bus voltage flexibility on the interface bus.
- Uses flow control.
- Increases complexity of the circuit when number of slaves and masters increases.

XI. CONCLUSION AND FUTURE SCOPE

I2C Single Master is successfully designed, simulated and synthesized. It can be used with multi slaves. This project can be further extended to design for multiple masters protocol. Multi-master communication protocol means two and more masters are used to communicate to support the many external device.

REFERENCES

[1] Philips Semiconductor , "I2C Bus Specification" version 2. 1, January 2000
[2] Philips Semiconductor, "I2C Manual", March 2003.
[3] Philips Semiconductor , "I2C-bus specification and user manual", April 2014.
[4] Texas Instrumentation, "Understanding I2C Bus", June 2015.
[5] Electronic Hub Tutorials, "Basics Of I2C Communication".
[6] Samir Palnitkar ,"Verilog HDL A guide to Digital Design and Synthesis"
[7] Git Hub, "Verilog guidelines".
[8] Xilinx – "Synthesis and Simulation Guide", UG626 (v 11.4) December 2, 2009
[9] Mentor Graphics, "ModelSim® User's Manual" ,Software Version 10.1c
[10] Tripti Singh "Design and Simulation of I2C protocol" International Journal for Research in Applied Science and Engineering Technology (IJRASET), VOLUME 2 issue XII, December 2014 ISSN: 2321-9653