Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.017

IJCSMC, Vol. 7, Issue. 6, June 2018, pg.64 – 69

An Efficient Design and Verification of I2C Master Core

Shruthi Solabappa Mugali[1], Dr. Meghana Kulkarni[2]
[1]VLSI design and Embedded System, Visvesvaraya Technological University, PG centre, Belgavi, India
[2]Associate professor under the dept. of VLSI Design and ES, Visvesvaraya Technological University, PG centre, Belgavi, India
[1] shrutismugali1994@gmail.com; [2] meghanak@vtu.ac.in

Abstract-In the current age of technology advancement is necessary to design any circuits. Before completing any tape out process in industry design, testing, fabrication and packaging are the most important steps to be considered. If the conditions are not met, then it cause a functional failure or huge loss for the company. So verification plays an important role. In conventional papers they used C, VHDL, Verilog and SystemVerilog for designing I2C master. Here also we used Verilog and SystemVerilog for designing I2C master. For simulation Synopsys VCS compiler tool is used. Our interface design includes read and write operations and will be able to communicate to master and slave through the I2C. The verification environment has been developed and test cases have been implemented for I2C Design Under Test (DUT). The verification has been done using System Verilog Hardware Description and Verification Language.

Keywords-I2C, Verilog, SystemVerilog, HDL, HVL

I. INTRODUCTION

The serial communication bus requires fewer IC connection pins, less wiring and the less number of traces on printed circuit boards. Many embedded system peripherals like analog-to-digital and digital-to-analog convertors, LCDs, and temperature sensors supports serial interface. Serial interface allow processors to communicate without the need for shared memory and the problems these can create. There are Serial

© 2018, IJCSMC All Rights Reserved

communication protocols like UART, CAN, USB, SPI, RS-232, RS-422, RS-485 for interfacing high speed and low speed peripherals. These require more pin connections and signals to connect devices [1]. To overcome this problem, the I2C protocol was introduced by Phillips which requires only two I/O lines for communication [2]. USB, SPI and UARTS all are one type to point type protocol. USB uses multiplexers to communicate with other device. Among these only I2C and CAN protocol uses software addressing and I2C is very efficient, easy to maintain and simple to design [3]. I2C is a simple multi-master, low-bandwidth, short-distance protocol [4].

It was created by Philips Semiconductors in the early 1980s. I2C was created to reduce the manufacturing cost of the electronic products as it provides minimum trace on board. Many semiconductor vendors support I2C-compatible devices in embedded systems including EEPROM, temperature current sensors, microcontrollers and real time clocks. The I2C allows 7-bit or 10 bit addressing using only two bi-directional lines: serial clock (SCL) and serial data (SDA). The pull-up resistors are only added for each of the lines. Each of the connected devices can act as a master or a slave device. The clock line can be drived by master device only. The transmission of 8-bit of data and 7-bit of address with a control bit is done serially using the interface. The device that initiates the transmission on the bus is usually known as the master, while all the other devices on bus are known as slave.

II. METHODOLOGY

Data Validity

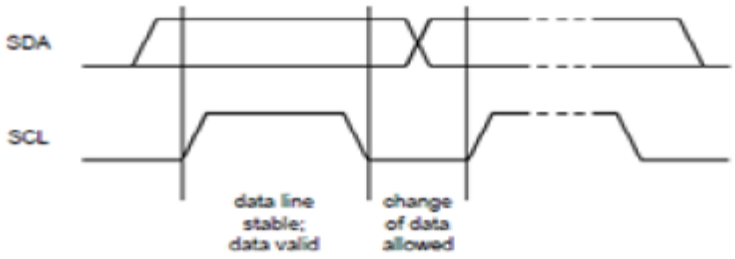

Fig 1: Bit transfer in the I2C bus.

In Figure 2.1 in order to transfer the data bit, at the high (1) period of the clock data is stable on the SDA line. The high (1) or low (0) state of the data line can only change when the SCL line is low (0). For each data bit transformation one clock pulse is generated.

Start And Stop Conditions

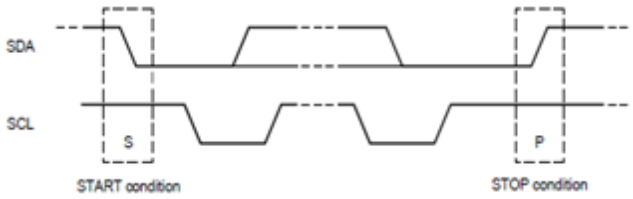

Fig 2: Start and Stop condition.

In any kind of circuit transactions begin with a START condition and ends by a STOP condition. When SCL is HIGH and SDA falls from HIGH to LOW gives a START condition. When SCL is HIGH and SDA rises from LOW to HIGH gives a STOP condition. These condition are always generated by the master.

© 2018, IJCSMC All Rights Reserved

Byte Format

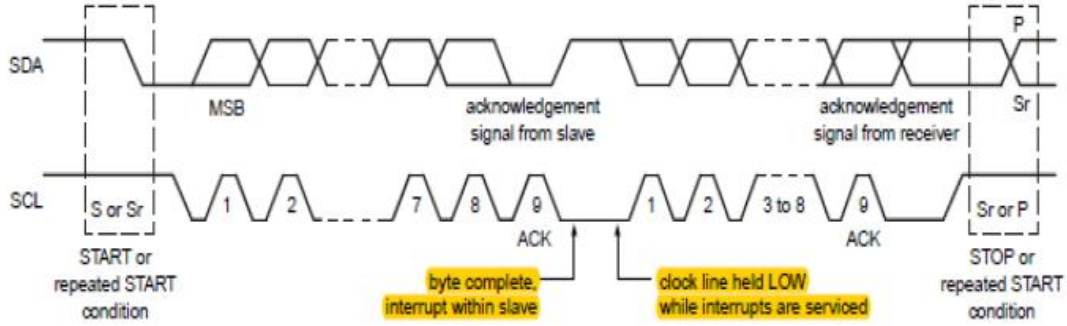

Fig 3: Data Transfer.

Every byte on the SDA line must be eight bits long. There is no limit for transfer of data over the number of bytes that can be transmitted. At transmission each byte must be followed by an Acknowledgement bit. With the Most Significant Bit (MSB) the data is transferred first . If a slave cannot receive or transmit another complete byte of data until it has performed some other function, it can hold the clock line SCL low(0) to set the master into a wait state. Data transfer then continues when the slave is ready for another byte of data and releases clock line SCL.

ACKNOWLEDGE (ACK) AND NOT ACKNOWLEDGE (NACK)

In order to avoid data loss the acknowledgement bit takes place after every byte transmission. This bit allows the slave signal in response with the master signal. All clock pulses are generated by master, including the acknowledgement bit which is in the ninth clock pulse. When SDA remains HIGH during this ninth clock pulse, this is defined as the Not Acknowledge signal. The master can then generate either a STOP condition to terminate the transfer, or a repeated START condition to start a new transfer.

III. DESIGN & IMPLEMENTATION OF I2C MASTER

Algorithm

State1: An idle condition: I2C bus doesn‘t perform any operation. (SCL and SDA remains high).

State 2:Start condition: Master initiates data transmission by providing START (SCL is high and SDA is from high to low).

State 3: Slave address -write: Master sends the slave address-write (11010000) to the slave.

State 4:If the slave address matches with the slave, it sends an acknowledgement bit in response to the master.

State 5:8 Bit Register Address will be transmitted to the slave. Again acknowledgement is sent to the master by the slave.

State 6: Data to be transmitted is sent to the slave by the master. After receiving the data, slave acknowledges the master.

State 7:Stop condition: Slave sends a stop bit to the master to terminate the communication (SCL is high and SDA is from Low to high). For performing read operation, write operation is performed first and then read operation is done. Slave address for read is 11010001. (State 7 will not be performed for read operation)

State 8:Master transmits slave address for read operation to the slave.

State 9:Master receives the data from the slave and acknowledges the slave.

State 10: Master sends a STOP bit to terminate the connection (SCL is high and SDA is from Low to high).

Based on the working of I2C master algorithm is written, such a way design is done for Verilog and SystemVerilog.

Figure 4 and Figure 5 represents the schematic diagram of I2C Mater using Verilog and SystemVerilog.

© 2018, IJCSMC All Rights Reserved

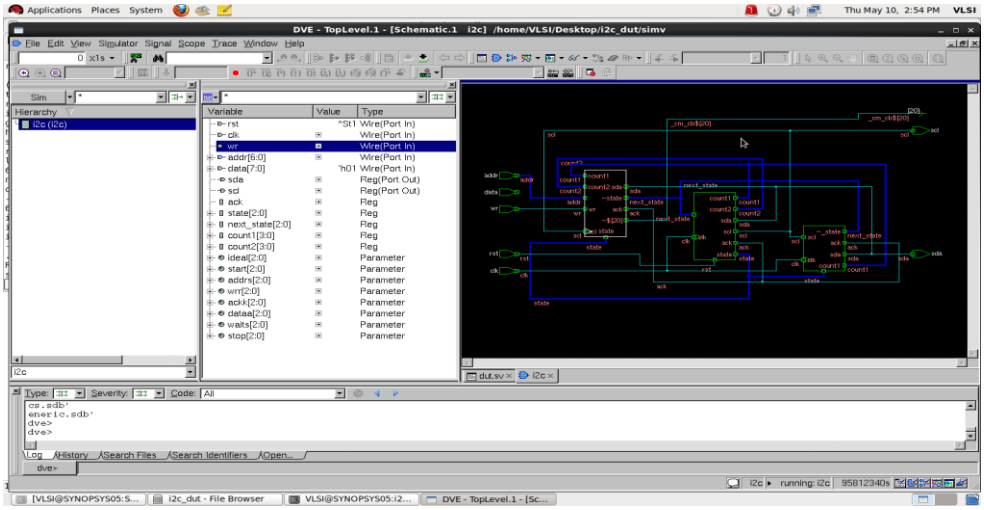

Fig 4: Proposed schematic for I2C Master using Verilog

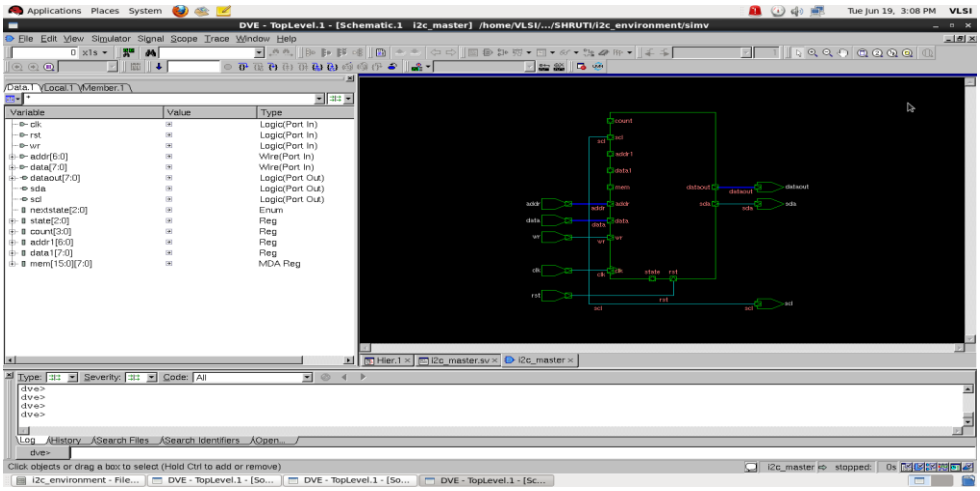

Fig 5: Proposed schematic for I2C Master using SystemVerilog

IV. EXPERIMENTAL RESULT

Output wave forms for proposed I2C master using Verilog are shown in Fig 6 and Fig 7. Results for read and write operations are observed.

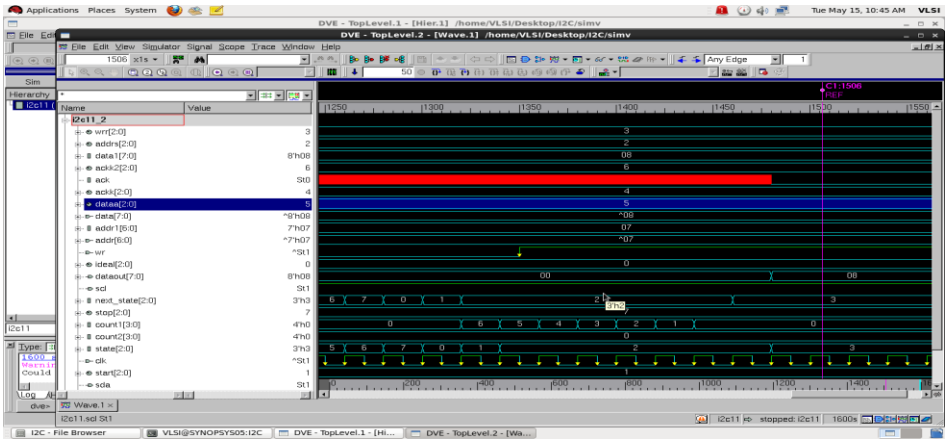

Fig 6: Output Waveform of proposed Verilog for read operation

© 2018, IJCSMC All Rights Reserved

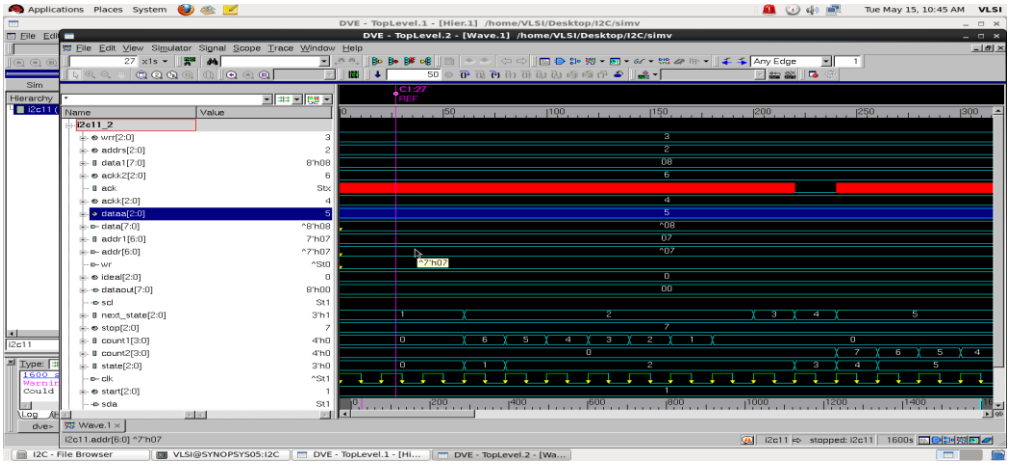

Fig 7: Output Waveform of proposed Verilog for write operation

Output wave forms for proposed I2C master using SystemVerilog are shown in Fig 8 and Fig 9. Results for read and write operations are observed.

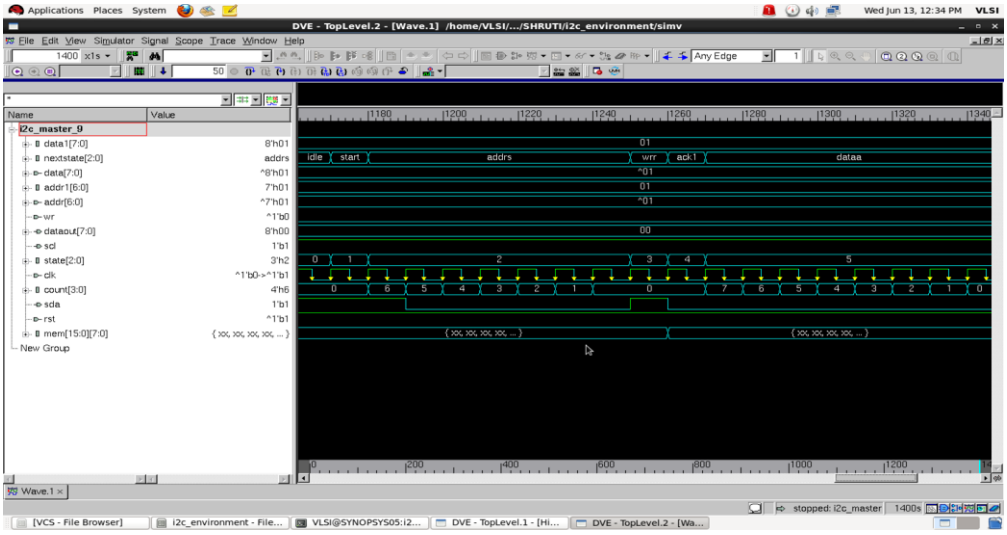

Fig 8: Output Waveform of proposed SystemVerilog for write operation

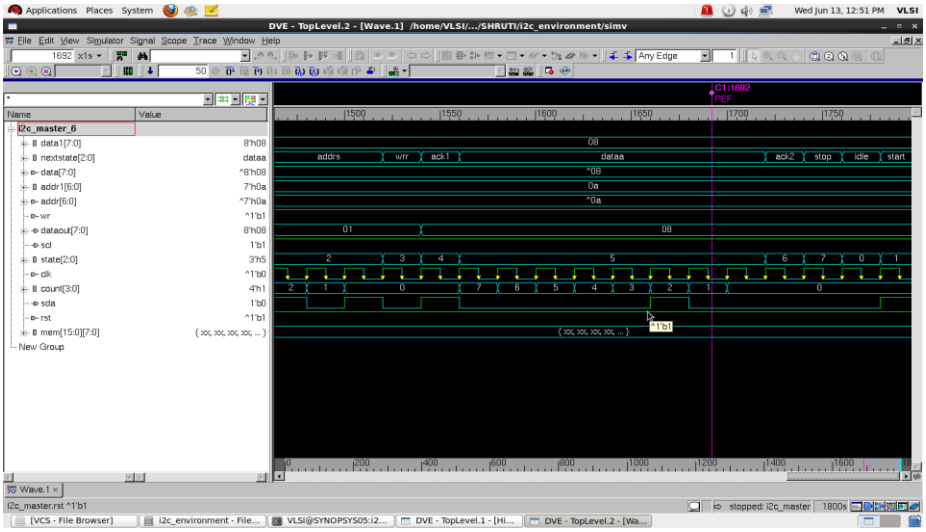

Fig 9: Output Waveform of proposed SystemVerilog for read operation

© 2018, IJCSMC All Rights Reserved

V. CONCLUSION

The ideal I2C bus has high performance, low cost, flexibility, easy upgradability. I2C bus controller is designed in Verilog and SystemVerilog simulated in Synopsys VCS. The working of I2C is software addressing based which can attach large number of devices by using only two lines SDA and SCL and pull up resistors. Any low speed peripheral devices can be interfaced using I2C bus protocol as master. It provides the efficient data transfer and synchronization by using acknowledgement and clock stretching. This can be implemented as a unit in a network that contains multiple masters and slaves to coordinate the entire system by clock synchronization techniques.

REFERENCES

[1] Subham Punit Patro, Shreela Dattawadkar, Shantala Kulkarni, Dr.Kiran Bailey, and Sunil Matange, "Verification of I2C module for Multiprotocol Serial Controller," International Journal of Current Trends in Engineering & Research (IJCTER), vol.2, Issue 4, pp. 548 – 555, April 2016

[2] Bollam Eswari, N.Ponmagal, K.Preethi, S.G.Sreejeesh "Implementation of I2C Master Bus Controller on FPGA" International conference on Communication and Signal Processing, - IEEE- 978-1-4673-4866-9/13, April 3-5, 2014.

[3] J. J Patel, Prof B. H. Soni "Design and implementation of I2C bus controller using Verilog" Journal of Information Knowledge and Research in Electronics and Communication Engineering ISSN: 0975– 6779, Volume–02, Issue-02, pp 520-522 Nov. 2012.

[4] Shoaib. Shah Sobhan, Sudipta. Das and Iqbalur. Rahman "Implementation of I2C using System Verilog and FPGA" International Conference on Advancements in Electronics and Power Engineering (ICAEPE'2011), Bangkok, Dec 2011.

[5] I2C-bus specification and user manual, Rev, vol. 3, p. 19, National Semiconductors, 2007.

© 2018, IJCSMC All Rights Reserved