OPEN ACCESS

International
Journal
Of Modern Engineering Research (IJMER)

An Implementation of I2C Slave Interface using Verilog HDL

Deepa Kaith[1], Dr. Janankkumar B. Patel[2], Mr. Neeraj Gupta[3]
1(Student, M. Tech. Electronics and Communication, Amity University Haryana, India)
2(Professor, Electronics and Communication, Amity University Haryana, India)
3(Assistant Professor, Electronics and Communication, Amity University Haryana, India)

ABSTRACT: *The focus of this paper is on implementation of Inter Integrated Circuit (I2C) protocol following slave module for no data loss. In this paper, the principle and the operation of I2C bus protocol will be introduced. It follows the I2C specification to provide device addressing, read/write operation and an acknowledgement. The programmable nature of device provide users with the flexibility of configuring the I2C slave device to any legal slave address to avoid the slave address collision on an I2C bus with multiple slave devices. This paper demonstrates how I2C Master controller transmits and receives data to and from the Slave with proper synchronization.*
The module is designed in Verilog and simulated in ModelSim. The design is also synthesized in Xilinx XST 14.1. This module acts as a slave for the microprocessor which can be customized for no data loss.
Keywords: *Inter Integrated circuit, serial data, serial clock, slave, verilog*

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 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 [8]. To overcome this problem, the I2C protocol was introduced by Phillips which requires only two I/O lines for communication [1]. 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 [9].

I2C is a simple multi-master, low-bandwidth, short-distance protocol [6]. It was developed 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. I2C PROTOCOL

I2C provides chip-to-chip communication using only two lines in the interface so ICs can communicate with fewer pins. The two wires in the I2C bus carry 7 bit address, R/W control bit, and data bit. The data (SDA) wire carries the data, while the Clock (SCL) wire synchronizes the sender and receiver during the transfer. Devices that use the I2C bus can perform the same function with few pins as their larger parallel interface equivalents. The I2C bus has three modes of operation: standard mode (0 to 100 kbps), fast mode (0 to 400 kbps) and high-speed mode (0 to 3.4 Mbps) [1][10].

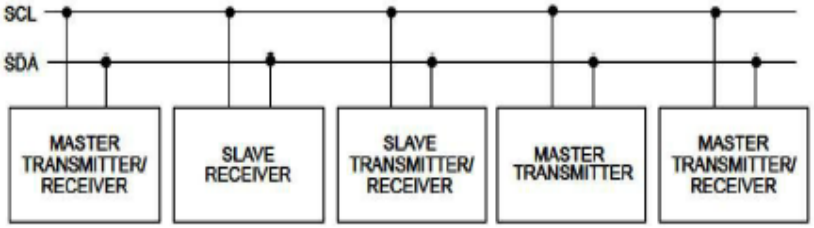

Fig.1: I2C bus configuration using masters and slaves

2.1 I2C Protocol:

The I2C bus physically consists of two bidirectional active lines- **s**erial Data line, **s**erial Clock line and a ground connection. Each device connected to the bus has its own unique address and can act either as a receiver or a transmitter, depending on the required functionality [8].

Master will issue an 'Attention' signal to all of the connected devices known as START. Then the Master sends the ADDRESS of the device it wants to communicate with and a Read or a Write bit. All devices compare it with their own address and the one whose address is matched will produce a response signal as an Acknowledgement. If it doesn't match, they simply wait until the bus is released. On receiving acknowledgement, master can start transmitting or receiving DATA. The transferred data is 8 bits long followed by an acknowledgement bit which is repeated till whole transmission takes place. When all is done, the Master will issue the STOP condition.

START (S)	Slave Address	R/W	ACK (A)	DATA	ACK	DATA	ACK (A)	STOP (P)

Fig.2: I2C Communication Frame Format

In this design there is a single slave which takes the command from the master, takes the bus control and performs the read and write operation accordingly. For synchronization purpose, the clocking stretching is done by the slave module which helps the slow peripherals to communicate with the fast devices.

2.2 I2C Bus Terminology

Transmitter: The device which sends the data or message to the receiver by the bus.
Receiver: The device which receives the data or message from the bus.
Master: It is the device which generates clock signals, initiates a transfer i.e. start condition and terminates a transfer i.e. stop condition. It may also act as a transmitter or a receiver.
Slave: It is the device addressed by a master. Also it may act as a transmitter or a receiver.
Multi master: More than one master can attempt to control the bus at the same time without demeaning the data.
Arbitration: It is done to ensure that only one master controls the bus if more than one master simultaneously tries to do so and thus avoid corruption of the data.
Synchronization: It is a procedure used to synchronize the clock signals of two or more devices.

2.3 General Characteristics

2.3.1 Write operation: It is done when the R/W bit is 0.

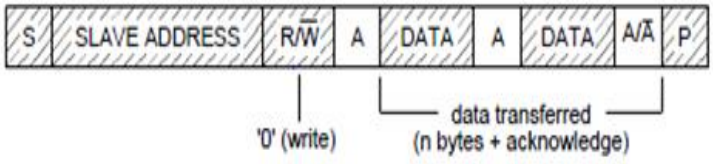

Fig.3: Write operation frame format

2.3.2 Read operation: It is done when the R/W bit is 1.

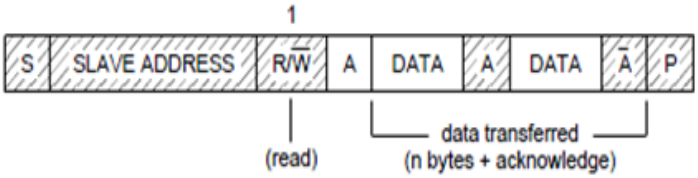

Fig.4: Read operation frame format

2.3.3 START and STOP conditions:
All communication begins from a START (S) and can be terminated by a STOP (P) signal. While SCL being High, a high to Low transition on the SDA line depicts a START condition and a low to High transition on the SDA line depicts a STOP condition [1]

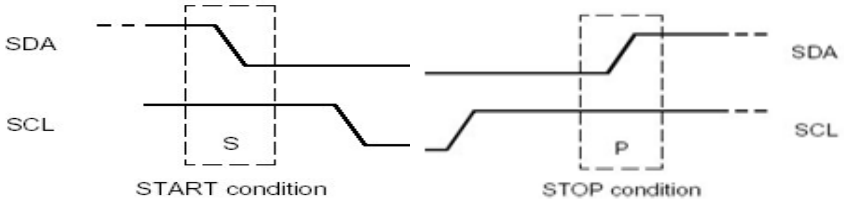

Fig.5: Start and Stop Condition

2.3.4 Transfer Data Byte format: every data byte must be 8 bit long to be put on the SDA line on the negative level. Each byte is followed by an Acknowledge bit which is done by pulling down the sda line.

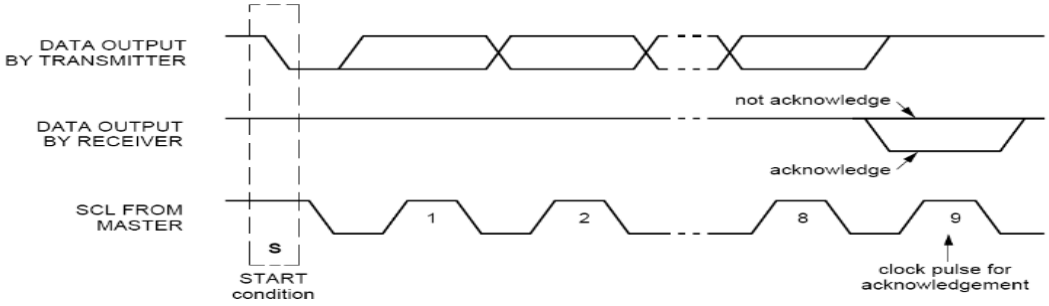

Fig.6: Acknowledgements in I2C protocol

2.3.5 Data Validity: The data on the SDA line are valid for the high period of clock pulse. It is shown in the figure below.

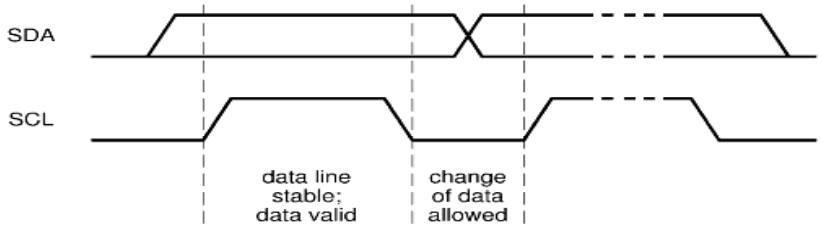

Fig.7: Timings of data validation

2.3.6 Transmitting a byte to a slave device
After start condition, a byte transmitted will identify the slave on the bus and will select the mode of operation. If the R/W bit in the address was set to 0, transmission of byte is done.

2.3.7 Receiving a byte from a slave device

Once the slave has been addressed and the slave has acknowledged this, a byte can be received from the slave if the R/W bit in the address was set to 1(READ).

III. Operation Algorithm

1. Master generates a START signal and all the slave devices listen to the bus.
2. Master sends a slave device ADDRESS + R/W bit.
3. Slave sends an ACKNOWLEDGE bit to the master if address is matched.
4. R/W bit is checked and the direction of data transfer is decided.
5. Slave sends 8-bit DATA to the master when R/W bit is 1 and receives 8-bit DATA from the master when R/W bit is 0.
6. An ACKNOWLEDGE signal is send after receiving the each 8-bit DATA by the receiver unit.
7. The DATA is repeatedly send or receive if there is no STOP signal.
8. The Master sends a STOP signal if data transmission is done.

IV. Functional Description Of I2c Slave

The I2C slave module designed here supports the major features of I2C specification. It responds to the commands issued by an I2C master. The I2C slave acknowledges the successful receive of address or data by pulling down the sda line during the ACK stage. This provides the handshaking mechanism to the connecting devices. When a slave cannot process the information fast enough it informs the master using few techniques. The first is slave simply does not respond during the ACK stage which causes the master to issue a stop command and restart transmission. The second is slave can use a clock stretching technique by pulling down the scl line as long as it needs. This helps in timing synchronization. The master will wait until the scl is released and continue to complete the previous transmission.

4.1 Input/ Output Connections:

The various input and output connections for the I2C component are below.

sda – in/out: This is the I2C data signal. It is a bi-directional data signal used to transmit or receive all bus data.

scl – in/out: This is the master generated I2C clock signal. The slave although never generates the clock signal, but it may hold it low until it is ready to NAK or ACK the latest data or address.

4.2 Functional diagram of I2C Slave:

I2C module provides interface to I2C bus and the peripheral devices that are connected to the bus. SCL and SDA here are bidirectional lines. The functional diagram is shown in the figure below

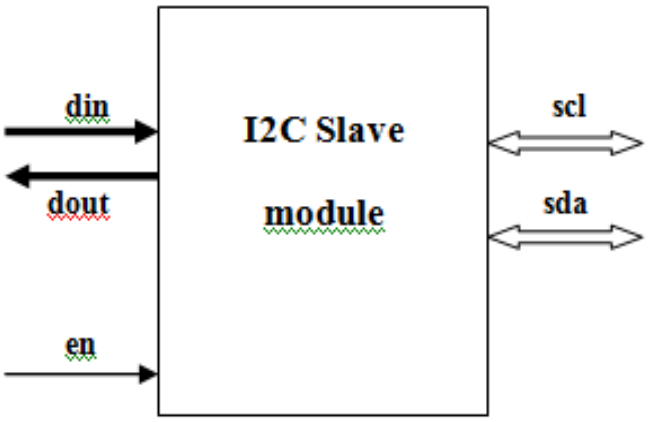

Fig.8: I2C Slave functional diagram

V. Simulation And Result

The result obtained after simulation using ModelSim is shown below. Here, after matching the address of slave, read or write operation with the desired slave is performed accordingly.

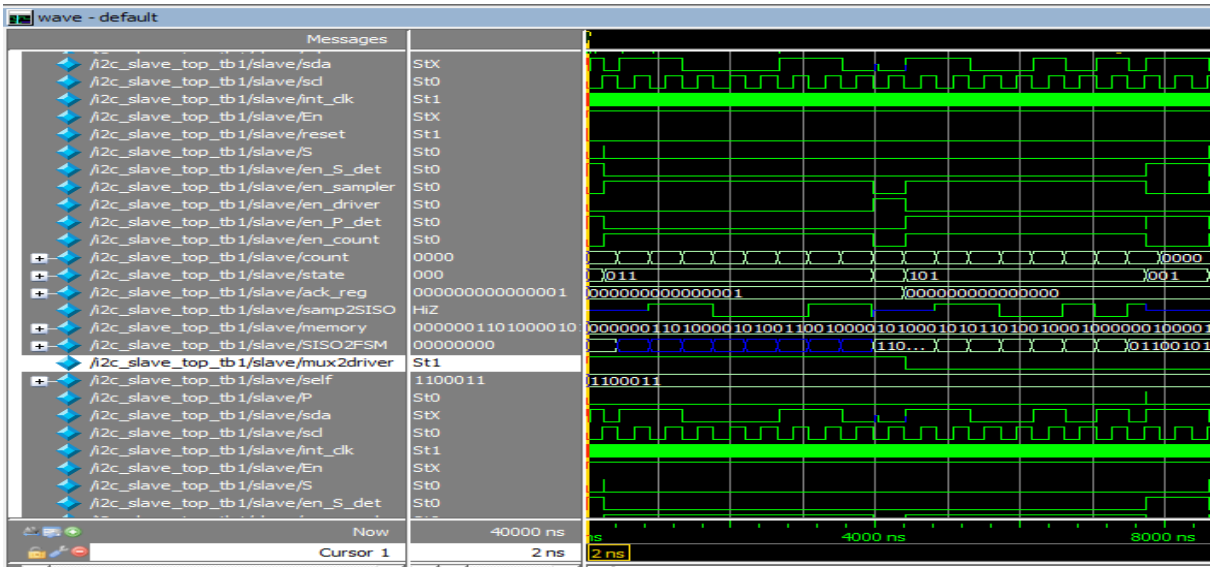

Fig.9: Output waveforms showing read operation

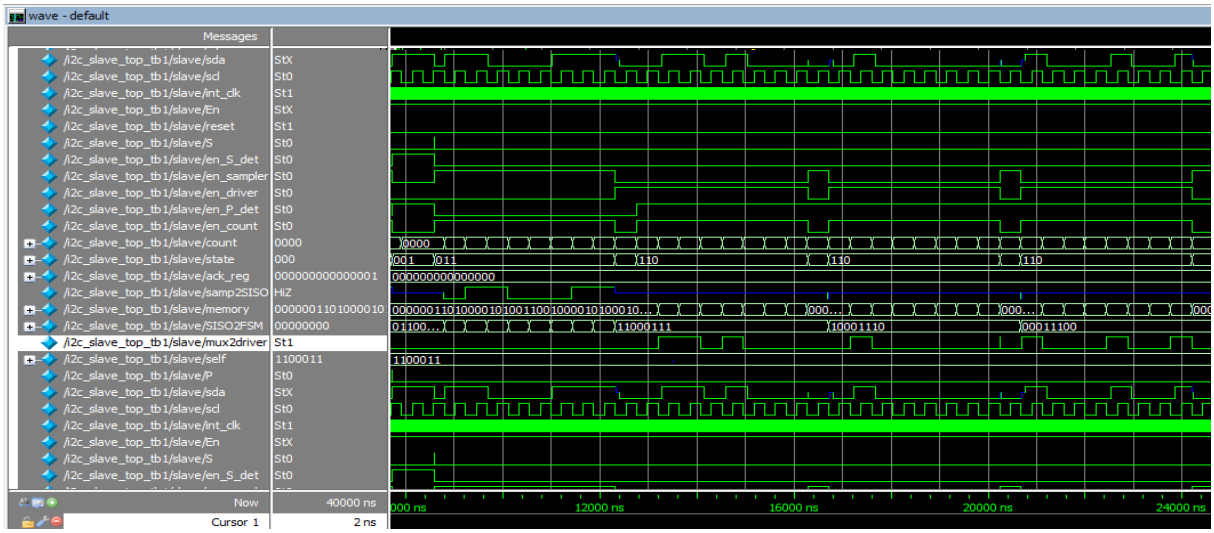

Fig.10: Output waveforms showing repeated write operation

The Resistor-transistor logic diagram viewed from XILINX 14.1 is shown in figure below

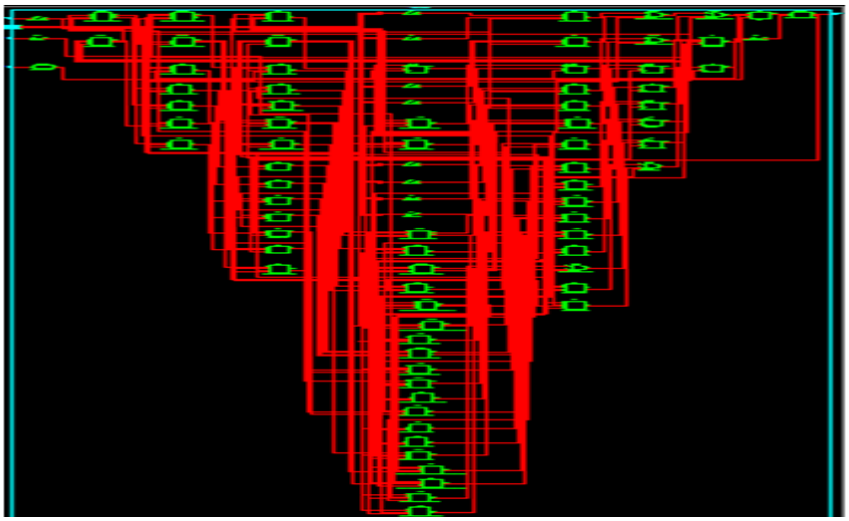

Fig.11. Technology tree schematic of the I2C Slave

VI. Conclusion And Scope

The ideal I2C bus has high performance, low cost, flexibility, easy upgradability. I2C bus controller is designed in verilog and simulated in ModelSIM. 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] Philips Semiconductor, "I2C-bus specification and user manual", version 2.1 January 2000.
[2] J.M. Irazabel & S. Blozis, Philips Semiconductors, "I2CManual, Application Note, ref. AN10216-0" March 24, 2003.
[3] Richard Herveille, "I2C Master Core Specification" Open Cores,2003
[4] Xilinx "Spartan-3A/3AN FPGA Starter Kit Board User Guide",version 1.1,2008.
[5] F.Leens, "An Introduction to I2C and SPI Protocols", IEEE Instrumentation & Measurement Magazine, pp. 8-13, February 2009.
[6] 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.
[7] 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.
[8] 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, 2013.
[9] Jayant Mankar, Chaitali Darode , Komal Trivedi "Review of I2C protocol" International Journal of Research in Advent Technology, E-ISSN: 2321–9637 Volume 2, Issue 1, January 2014.
[10] Tatiana Leal-del Río, Gustavo Juarez-Gracia, L. Noé Oliva-Moreno "Implementation of the communication protocols SPI and I2C using a FPGA by the HDL-Verilog language" Research in Computing Science 75 pp. 31–41; July, 2014