Designing of I2C Master based on Speed Modes

Ms. Sowmyashree B G
PG student
Dept. of E&C, UBDTCE
Davangere, India

Mr. Arun Raj S R
Assistant Professor
Dept. of E&C, UBDTCE
Davangere, India

***Abstract*:- This paper focus on the efficient designing and simulating of Inter Integrated Circuit (I2C) master controller based on speed modes. The design follows I2C specifications for address sending and data transfer operations. The design is software based that provides easy up-gradation of the whole electronic system to ensure no data loss. The master controller is designed in Verilog and is simulated using Xilinx_ISE_DS_ 14.6.**

Keywords- Inter integrated circuit, master, slave, speed modes.

I. INTRODUCTION

The I2C (Inter-Integrated circuit), pronounced as I2C bus has become the de-facto world standard that is now used in different ICs. The collisions are prevented using the wired-and configurations of the serial data (SDA) line and the serial clock (SCL), and data loss is prevented by the fact that every byte on the SDA line has to be followed by an acknowledge. This protocol can support multiple masters, that provide a simple and efficient method of data exchange between devices and is used for faster devices to communicate with slower devices and each other. Originally, I2C bus was limited to the speed of 100 kbps. Now it has been upgraded for different speed modes that provide efficient data transmission.

II. PREVIOUS WORK

In the world of serial data communication, there are protocols like RS-232, RS-422, RS-485, SPI (Serial peripheral interface), and Microwire for interfacing high speed and low speed peripherals. These protocols require more pin connection in the IC(Integrated Circuit) for serial data communication to take place, as the physical size of IC have decreased over the years, we require less amount of pin connection for serial data transfer. USB/SPI/Microwire and mostly UARTS are all just 'one point to one point' data transfer bus systems. They use multiplexing of the data path and forwarding of messages to service multiple devices. To overcome this problem, the I2C protocol was introduced by Phillips, which requires only two lines for communication with two or more chips and can control a network of device chips with just a two general purpose I/O pins whereas, other bus protocols require more pins and signals to connect devices.

III. I2C PROTOCOL

The I2C-bus supports any IC fabrication process (NMOS, CMOS, bipolar). It consists of two active wires and a ground connection. The active wires, called serial data (SDA) and serial clock (SCL), are both bidirectional they carry information between the devices connected to the bus as shown in fig.1. Each device is recognised by a unique address whether it's a microcontroller, LCD driver, memory or keyboard interface and can operate as either a transmitter or receiver, depending on the function of the device. Obviously an LCD driver is only a receiver, whereas a memory can both receive and transmit data. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers. A master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave.

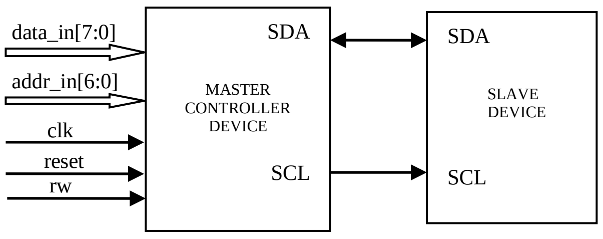


Fig.1: RTL view of I2C bus architecture

A. I2C FEATURES

I2C requires only two bus lines; a serial data line (SDA) and a serial clock line (SCL). Devices that are connected to the bus are software addressable with unique addresses. It is a true multi-master bus which overcomes collision detection and arbitration to prevent data corruption if two or more masters simultaneously initiate data transfer. 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 Fast-mode Plus. Serial, 8-bit oriented, unidirectional data transfers up to 5 Mbit/s in Ultra Fast-mode. On-chip filtering rejects spikes on the bus data line to preserve data integrity. The number of Ics that can be connected to the same bus is limited only by a maximum bus capacitance. Generation of clock signals in the I2C bus is always the responsibility of the master devices. One clock pulse is generated for each data bit transferred.

B. TERMINOLOGIES

- *Transmitter:* The device which sends data to the bus;
- *Receiver:* The device which receives data from the bus;
- *Master:* The device which initiates a transfer generates clock signals and terminates a transfer;
- *Slave:* The device addressed by a master;
- *Multi-master:* More than one master can attempt to control the bus at the same time without corrupting the message;
- *Arbitration:* Procedure to ensure that, if more than one master simultaneously tries to control the bus,

only one is allowed to do so and the winning message is not corrupted.

- *Synchronization:* Procedure to synchronize the clock signals of two or more devices.

C. SPECIFICATIONS

1. *Start (s) and stop (p):* These are used to initiate and stop transactions on the I2C-bus. START is generated by pulling the SDA line low while SCL is high and a STOP is generated by pulling the SDA line high while SCL is high as shown below in fig.2. It is possible to issue repeated starts, initiating a new communication sequence without relinquishing control of the bus to other master.

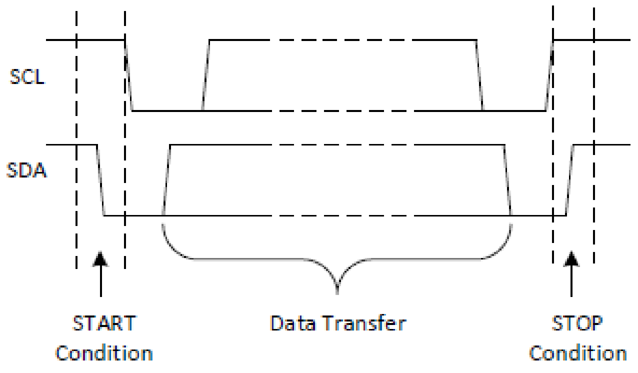


Fig.2: start and stop condition

2. *Data validity:* The master will simply continue generating clock pulses at a regular interval, and the data will be placed on SDA by either the master or the slave, depending on whether the RW bit indicated a read or write operation. When SCL is HIGH data must be stable and can change when SCL is LOW as shown in fig.3.

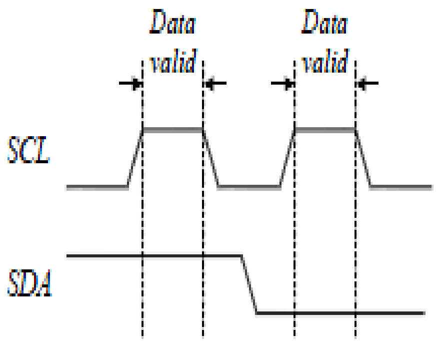


Fig.3: data validation timings

3. *Acknowledgements*: It is given by the receiver by pulling down (LOW) the SDA line after transmitter releases the bus. If the SDA remains HIGH during this clock pulse, it is known as Not Acknowledge (NACK) signal. After NACK, the master can either generate a repeated START for a new transfer or a STOP to abort the data transfer.

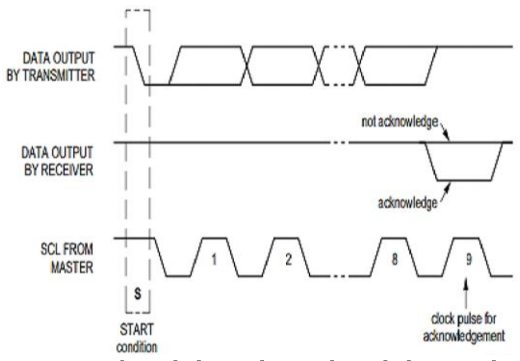


Fig.4: acknowledge and not acknowledge signals

4. *Write or read:* The communication frame for write and read is shown in fig.5. The control bit (rw) sent along with the address bit decides the direction of transfer.

☐ master

▨ slave

(RW=0)WRITE

S	slave address	rw= 0	a c k	data	a c k	data	a c k	P

(RW=1)READ

S	slave address	rw= 1	a c k	data	a c k	data	a c k	P

Fig.5: I2C communication frames for write and read

IV. OPERATION

- Master initiates the data transfer by sending START signal. It is as an attention signal to all the slave devices on the bus.
- Start is followed by the 7-bit ADDRESS of the slave it wants to communicate with.
- ADDRESS is followed by a RW bit which acts as a control bit to decide the direction of data transfer.
- Master releases the bus and wait for the acknowledge (ACK) bit from the slave.
- The desired slave pulls the SDA line LOW to send ACK signal.
- Depending on RW bit value either master or slave sends the 8-bit data and acts as the transmitter.
- The other device acts as a receiver and sends ACK after receiving 8-bit data by pulling SDA LOW.
- After (N)ACK the master ends the transmission by sending STOP signal.
- Any master can now initiate a new data transfer by taking the control of bus.
- Along with this there will be 2-bit mode signal which describes the speed modes and based on

the speed, the clock gets divided and the information transfer takes place.

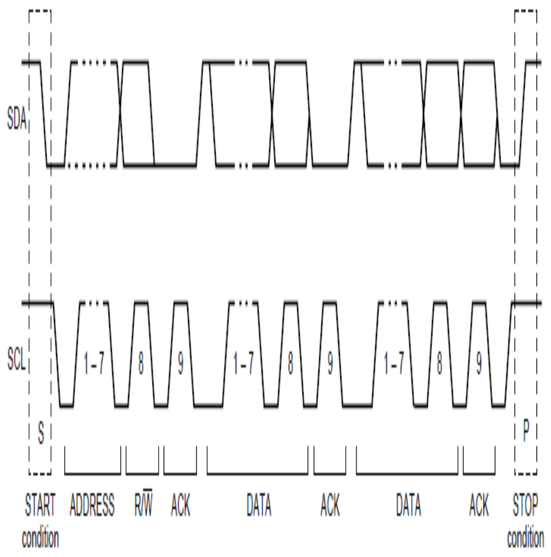


Fig.6: complete data transfer operation

A. *Efficient methods of I2C*

The various methods used to ensure that there is no bus conflict arising and thus, no data loss are:

1. *Clock synchronization:* All masters generate their own clock on the SCL line to transfer messages on the I2C-bus. Data is only valid during the HIGH period of the clock. A defined clock is therefore needed for the bit-by-bit arbitration procedure to take place.Clock synchronization is performed using the wired-AND connection of I2C interfaces to the SCL line. The SCL line will therefore be held LOW by the device with the longest LOW period and devices with shorter LOW periods enter a HIGH wait-state during this time.

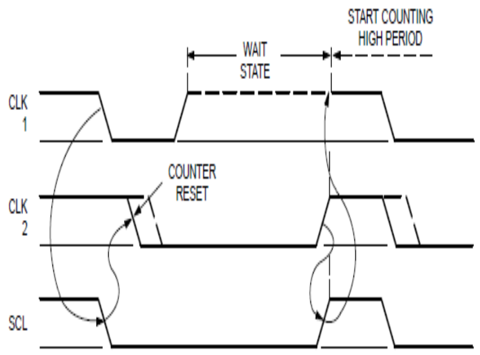


Fig.7: clock synchronization with multiple masters

2. *Arbitration:* This includes master only and is done on SDA line while SCL is HIGH. Arbitration can continue for many bits. Its first stage is comparison of the address bits. If the masters are trying to address the same device, arbitration continues with comparison of the data. Because address and data information on the I2C-bus is used for arbitration, no information is lost during this process. A master which loses the arbitration can generate clock pulses until the end of the byte in which it loses the arbitration.

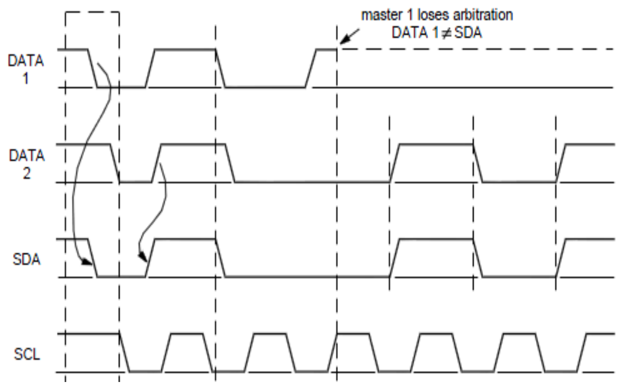


Fig.8: arbitration procedure

Arbitration isn't allowed between:

- A repeated START condition and a data bit.
- A STOP condition and a data bit.
- A repeated START condition and a STOP condition.

3. *Clock stretching:* This mechanism gives an advantage in communication with the slow peripheral devices. If the speed of the slave device is less than the clock generated by the master, it can hold the SCL line low to indicate that it is not ready yet. Then the master will wait till the clock is released by the slave before the next transfer.

B. *Speed modes of I2C*

There are 4 different speed modes that helps to faster the transfer rate and to reduce the data loss.

1. *Standard mode (SM):*
   The addressing procedure for the I2C-bus is such that the first byte after the START condition usually determines which slave will be selected by the master. In this mode the speed of data and address transferring will be limited to 100kbps.

2. *Fast mode (FM):*
   In the fast-mode of the I2C-bus, some changes are made along with the previous specifications, they are:

- The maximum bit rate is increased to 400 kbit/s.
- Timing of the serial data (SDA) and serial clock (SCL) signals has been adapted.
- The inputs of fast-mode devices must incorporate spike suppression and a Schmitt trigger at the SDA and SCL inputs
- The output buffers of fast-mode devices must incorporate slope control of the falling edges of the SDA and SCL signals.

- If the power supply to a fast-mode device is switched off, the SDA and SCL I/O pins must be floating so that they don't obstruct the bus lines.

3. *Fast mode plus (FM+):*
   - Fast-mode Plus (Fm+) devices offer an increase in I2C-bus transfer speeds and total bus capacitance.
   - Fm+ devices can transfer information at bit rates of up to 1 Mbit/s.
   - Fm+ devices also offer increased drive current allowing them to drive longer and/or more heavily loaded buses so that bus buffers do not need to be used.
   - The drivers in Fast-mode Plus parts are strong enough to satisfy the Fast-mode Plus timing specification with the same 400 pF.
   - Bus speed can be traded against load capacitance to increase the maximum capacitance by about a factor of ten.

4. *High speed mode (HS-M):*
   - Hs-mode devices can transfer information at bit rates of up to 3.4 Mbit/s.
   - No arbitration or clock synchronization is performed during Hs-mode transfer in multi-master systems, which speeds-up bit handling capabilities. The arbitration procedure always finishes after a preceding master code transmission in F/S-mode.
   - Hs-mode master devices generate a serial clock signal with a HIGH to LOW ratio of 1 to 2. This relieves the timing requirements for set-up and hold times.
   - Hs-mode master devices can have a built-in bridge. During Hs-mode transfer, the high-speed data (SDAH) and high-speed serial clock (SCLH) lines of Hs-mode devices are separated by this bridge from the SDA and SCL lines of F/S-mode devices. This reduces the capacitive load of the SDAH and SCLH lines resulting in faster rise and fall times.
   - The inputs of Hs-mode devices incorporate spike suppression and a Schmitt trigger at the SDAH and SCLH inputs.
   - The output buffers of Hs-mode devices incorporate slope control of the falling edges of the SDAH and SCLH signals.

V. RESULT

The I2C has been designed in Verilog and simulated using Xilinx_ISE_DS_14.6 to verify the design functioning. These designs works for both read and write cycle and is given below.

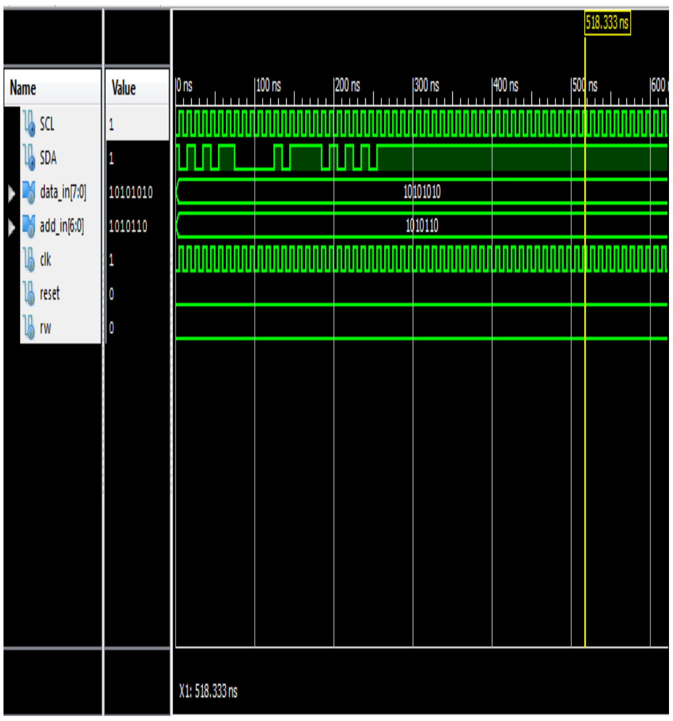


Fig.9: I2C write operation cycle

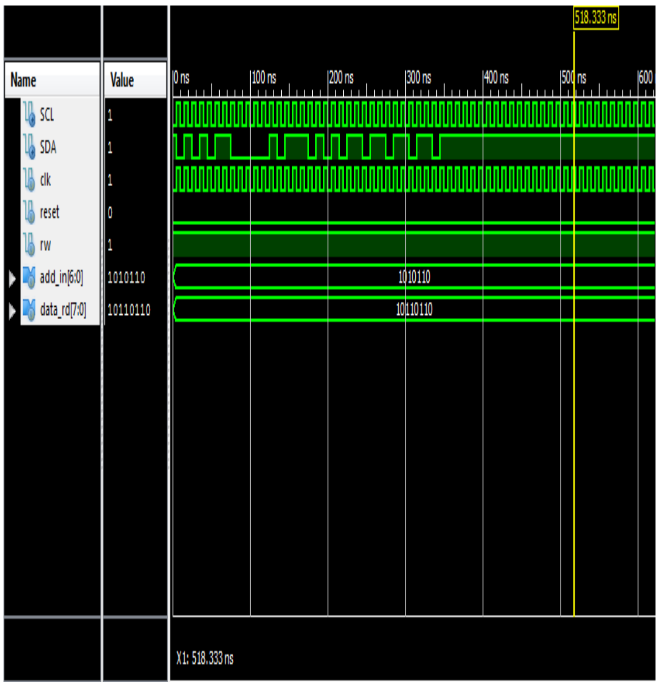


Fig.10: I2C read operation cycle

The technology tree schematic can be viewed as shown below in fig.11:

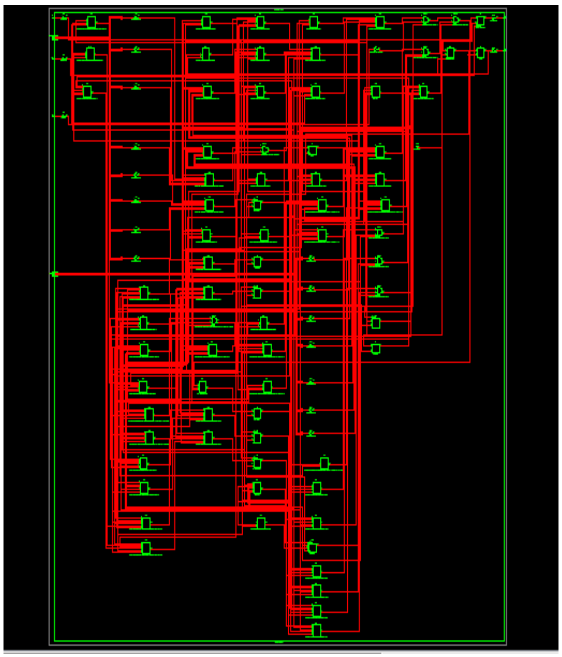
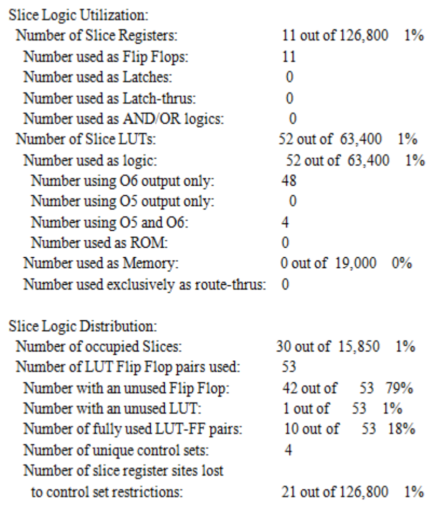


Fig.11: Technology schematic of I2C

Design utilization summary:



```
Slice Logic Utilization:
  Number of Slice Registers:                11 out of 126,800    1%
    Number used as Flip Flops:              11
    Number used as Latches:                  0
    Number used as Latch-thrus:              0
    Number used as AND/OR logics:            0
  Number of Slice LUTs:                     52 out of  63,400    1%
    Number used as logic:                   52 out of  63,400    1%
      Number using O6 output only:          48
      Number using O5 output only:           0
      Number using O5 and O6:                4
      Number used as ROM:                    0
    Number used as Memory:                   0 out of  19,000    0%
    Number used exclusively as route-thrus:  0

Slice Logic Distribution:
  Number of occupied Slices:                30 out of  15,850    1%
  Number of LUT Flip Flop pairs used:       53
    Number with an unused Flip Flop:        42 out of      53   79%
    Number with an unused LUT:               1 out of      53    1%
    Number of fully used LUT-FF pairs:      10 out of      53   18%
    Number of unique control sets:           4
    Number of slice register sites lost
      to control set restrictions:          21 out of 126,800    1%
```

VI. CONCLUSION

Inter integrated circuit master controller is successfully designed and simulated. As the number of devices connected to a system is going to increase, there is a need for a system which supports multiple protocols. This project mainly concentrates to reduce the system complexity and data loss. The logic synthesis tool will optimize the circuit in area and timing for the new technology.

REFERENCES

[1] Bollam Eswari, N.Ponmagal, K.Preethi, S.G.Sreejeesh ―Implementation of I2C Master Bus Controller on FPGA‖ in IEEE, International conference on Communication and Signal Processing, April 3-5, 2013.

[2] I2C Bus Specification, Philips Semiconductor, version 2.1, January2000.

[3] UM10204 I2C -bus specification and user manual Rev. 4 — 13 February 2012.

[4] IEEE Computer Society. IEEE Standard Verilog® Hardware Description Language, IEEE Std 1364-2001, The Institute of Electrical and Electronics Engineers, Inc,28 September 2001.

[5] Mr. J. J Patel, Prof B. H. Soni, ―Design And Implementation Of I2c Bus Controller Using Verilog‖ in Proc. Journal Of Information, Knowledge and Research in Electronics and Communication Engineering ISSN: 0975 – 6779, VOLUME – 02, ISSUE – 02, NOV 12 TO OCT 13.

[6] Ashwini s. Tadkal, Padmapriya Patil, ―DESIGN OF DUAL MASTER I2C BUS CONTROLLER‖ IJRET, Volume: 03,Special Issue:03, May-2014.