International Journal of Scientific Engineering and Applied Science (IJSEAS) - Volume-1, Issue-3, June 2015
ISSN: 2395-3470
www.ijseas.com

Design & Implementation of I^{2}C Protocol

Chinmay Modi[1], Heli Shah[2], Bhargav Tarpara[3]
[1, 2] *M.Tech., VLSI, U.V.Patel College of Engineering and Technology, Kherva, Mehsana, India*
[3] *Verification Technical Assistant, eiTRA, Ahmedabad, India*

***Abstract*—we present design of an intellectual property (IP) for inter-integrated circuit (I^{2}C) bus protocol. To enable multiple devices to communicate with each other over serial data bus without any data or address loss, as well as to enable faster devices with slower ones. The inter IC(I^{2}C) protocol was put forward by Philips semiconductors in 4th April 2014. This protocol design proposed for reusability concept. In this paper, a design on FPGA Platform is presented for I^{2}C protocol. Also proposed model is used for communication between multiple masters and multiple slaves. The entire design has been coded in verilog & verified using Spartan kit.**

Keywords— IP(Intellectual Property), I^{2}C(Inter Integrated Circuit), FPGA(Field Programmable Gate Array), Verilog, System Verilog(SV), Universal Verification methodology(UVM).

I. INTRODUCTION

THE I^{2}C-bus is a de facto world standard that is now implemented more than 10^3 different ICs manufactured by 50+ companies. Additionally, I^{2}C-bus is used in various control architectures like System Management Bus (SMBus), Power Management Bus (PMBus), Intelligent Platform Management Interface (IPMI), Display Data Channel (DDC) and Advanced Telecom Computing Architecture (ATCA)[1]. There are many similarities between seemingly unrelated design for various industries like consumer electronics, telecommunications and industrial electronics. For Example,

1. Some intelligent control in a single-chip micro-controller.
2. EEPROM, A/D & D/A Converter.
3. Audio-video systems, temperature sensor, Digital imaging.

To exploit these similarities and benefit to both systems designers and equipment manufacturers and to maximize hardware efficiency and circuit simplicity, that is why a simple bidirectional 2-wire bus for efficient inter-IC control. This bus is called the Inter IC or I^{2}C-bus. All I^{2}C-bus devices incorporate an on-chip interface which allows them to communicate directly with each other via the I^{2}C-bus. This design concept solves the many interfacing problems encountered when designing digital control circuits [1]. The model can be used as a master, multiple masters or as a slave, multiple slave or both. We first present the characteristics of the I^{2}C protocol and the controller, then focus on the modeling of the IP and show Simulation Results.

II. I^{2}C Bus Specification

The I2C bus is containing two-wire serial bus, one is for SDA (serial data) and other is for SCL (serial clock). Each device has its own unique address, and can work either as a transmitter or a as a receiver. The I2C master is the device that initiates a transfer and generates the clock for the same. Any device addressed by the master is also the slave. If more than one master attempts to transmit in address or data, there will be a collision. The I2C specification solves this collision by its arbitration process, clock synchronization and also clock Stretching concept. Before go to the arbitration process, Clock synchronization process. The basic I2C characteristics are given below. [2][3]

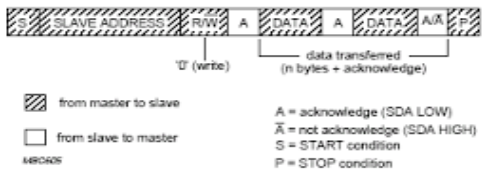
A. I^{2}C Characteristics

- Both Master and slave operation
- Unique Start/Stop/Repeated Start condition

- Slave selection protocol uses a 7-bit slave address
- Bidirectional data transfer
- Data transfer Speed
    1. standard mode - 100 Kbit/s
    2. fast mode - 400 Kbit/s
    3. High Speed mode - 3.4 Mbits/s
- Acknowledgement after each transferred byte
- No fixed length of transfer
- True Multi-master capability
- Fully supports Clock synchronization & arbitration process
- Data stability
- Programmable SCL frequency
- Soft reset of I2C Master/Salve
- Electrical & timing Specification[1]

B. I²C Addressing Format

The various devices on the I²C bus can be differentiated by their address. It contains 10 bit addressing mode. its 1st master send start/repeated start condition, 2nd step, master send 7 bit of particular slave device address out of many devices.3rd step, if bit indicate 1 for read and 0 for write bit respectively. Then 4thstep, acknowledgement bit indicate that slave device address received successfully.5th step, master sends data to slave device. 6th step master received acknowledgement and then master sends a stop condition that indicate no more data or address will there. Finally stop the transaction. [1][2]



A master-transmitter addressing a slave receiver with a 7-bit address. The transfer direction is not changed.

Figure 1: I²C Frame Format

C. Start & Stop Condition

Every data/address transfer in this protocol first requirement is start condition. START condition indicates that SCL must be high rising edge/state. And SDA line must follow the transition from High to Low. Initially both are at high impedance (Z) and for STOP condition indicate that SCL must be high rising edge/state. And SDA line must follow the transition from Low to High.[1][2]In software simulation based verification, the HDL code of the digital logic is simulated by the simulation software. Logic simulation is the primary tool used for verifying the logical correctness of a hardware design. In many cases, logic simulation is the first activity performed in the process of taking a hardware design from concept to realization. Test-bench can be written around the design under test (DUT) and inputs can be passed to the DUT through the test-bench. Simulation is completely generic and any hardware design can be simulated. Setup is simple, quick and easy highest level of controllability and observability Designer gets complete feedback of the verification process [2] [3].

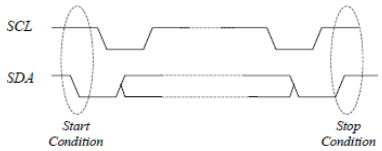


Figure 2: I2C Start & Stop Condition

D. Acknowledgement

Master/slave receivers pull data line low for one clock pulse after reception of a byte. Master receiver leaves data line high after receipt of the last byte requested. Slave receiver leaves data line high on the byte following the last byte it can accept.(Figure3).Receiver leaves data line high for one clock pulse after reception of a byte.(Figure4) [2]

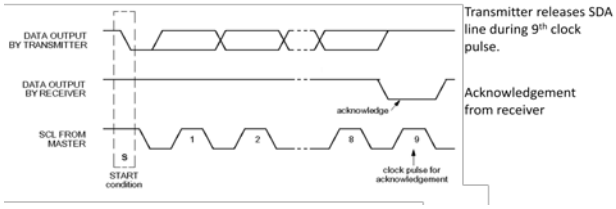


Figure 3: Acknowledgment Reception

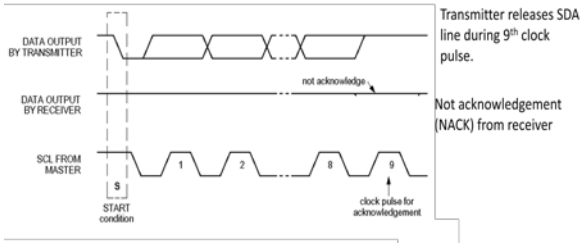


Figure 4: Negative Acknowledgement Reception

D. Data stability

Every transfer of data byte direction flow is MSB bit to LSB bit. Data should be consider valid if SCL state is high. So SDA signal must be remain stable during this period. [2]

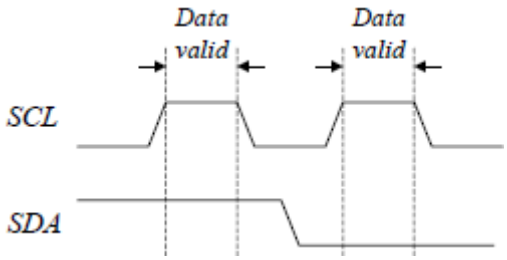


Figure 5: Data validity

E. Clock Synchronization

When two masters begin transmitting on a free bus at a same time then there must be a procedure to decide which takes control and which not. This is done by the clock synchronization and arbitration procedure.

In clock synchronization, wired-and connection can be performed in i2c interface to SCL line. During high to low transition on SCL line start counted off their low period of concerned master. If master clock goes LOW, Then it also holds the SCL line until the clock remains the HIGH state. However, if another clock will remain at its LOW period, then during the LOW to HIGH shift of the clock it may not change the state of the SCL line. So, master held the SCL line LOW with much long LOW period. After that the shorter LOW period enters the HIGH wait-state during this time by master.

When whole masters concerned about counted off their LOW period, so the clock line will be release and goes HIGH. Then, there is no difference between the master clocks & the state of the SCL line because all masters started counting their HIGH period. So, the first master to complete its HIGH period pulls the SCL line LOW again. In that way, a synchronized SCL clock will be generated with their LOW period determined by master with its longest clock LOW period, and their HIGH period determined by one at the shortest clock HIGH period.[1]

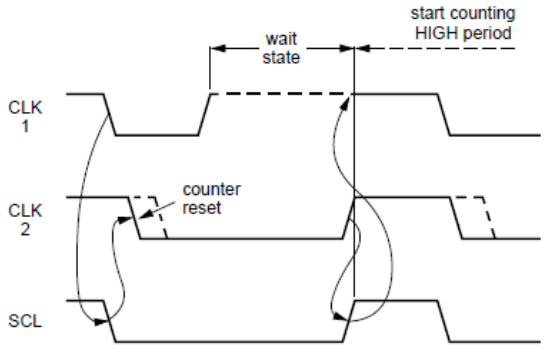


Figure 6: clock synchronization

F. Arbitration

In I^2C protocol, only master can involve in arbitration. Slaves are not involved in arbitration. Master initiate transfer only and only when bus is idle. Two masters may initiate a START condition within the minimum hold time of the I^2C bus START condition for which gives in a valid START condition on it. Arbitration is needed to check which master will complete their transmission as early as possible. There is no data lost in it. The advantage is that two master can complete its transmission without causing any hazard or error. Arbitration procedure keeps sending pattern bit by bit. When the SCL is gone high it will also verify SDA level for which is sent. For the first time a master try to initiates or generate a HIGH period, but that will be detecting the SDA level also gone LOW, the master knows that it will be lost the arbitration procedure and turns off its SDA output driver. Another master goes to finish their entire transaction. If a master also incorporates a slave function and it loses its arbitration during its address, there is possible that the possibility of winning masters may attempt to address the master. There will be losing master must be therefore switch over instantly to their slave mode. Figure shows the arbitration procedure for all the masters. For the

International Journal of Scientific Engineering and Applied Science (IJSEAS) - Volume-1, Issue-3, June 2015
ISSN: 2395-3470
www.ijseas.com

moments there can be dissimilarities between the internal data level of the master initiating DATA1 and also actual level on the SDA line, and the DATA1 output could be also switched off. That won't affect any of the data transfer initiated by the winning master. Since Final control of the I2C-bus is decided on address and data sent by true competing masters, there won't be centralize master inside it, nor any higher priority master given on the bus.[1][3]

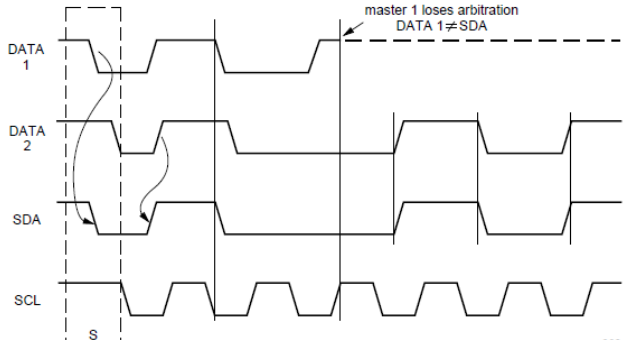


Figure 7: arbitration of two master

G. Clock stretching

Clock stretching is some time pause a transaction by holding the SCL line LOW at some duration. So the transaction won't continue until & unless the line will be released HIGH. Generally Clock stretching is usually not for multiple slaves. So it is optional and In fact, most of the slave devices are not include an SCL driver so they are not chances to stretch the clock. For example some device may send data at a faster rate but need more time to store it So slave then hold SCL line low after reception and receive acknowledgment of a byte to force the master enter into a wait state until particular slave is ready for the next byte of transfer in a type of handshake procedure. [1][4]

III. DESIGN & IMPLEMENTATION OF THE IP

In I^2C Protocol, we have been design master block and slave block. In I^2C protocol, Master can be work as a Slave same as Slave can be work as Master respectively. In Design, I have implemented format of the I^2C frame. I^2C.Master can generate a start and stop condition, send address or data towards slave. We can also receive acknowledgement from slave. Whole operation can perform in bidirectional way because Slave may be master or master Slave. Address and data will be generated on SDA line also correlating to the SCL line. All devices of (slaves) connected to the I^2C bus should be follow the wired- and condition. If the I2C bus is idle than it goes to high impedance state. Whether the BUS is 'high', the Master instantly captures the I2C bus by pulling down the BUS line to a 'low' state. whether two masters trying to get the bus at the same time, then the most winning one master control will be decided by following the I2C arbitration logic.[1] Functional block diagram is described as below.

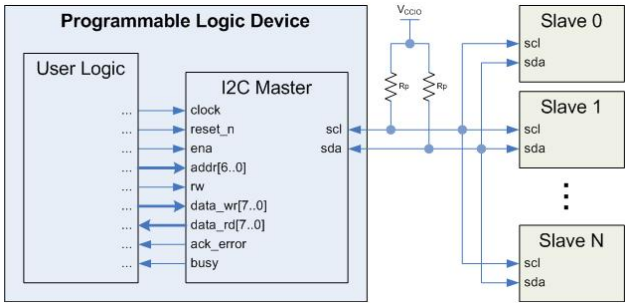


Figure 8: Functional Block Diagram

A. Master/Slave Top Level Block

Master basically Consist five Operations like initiator for start and stop condition, Address for particular Slave device, Write data to Slave vice versa, Read data from Slave vice versa, Clock generator.

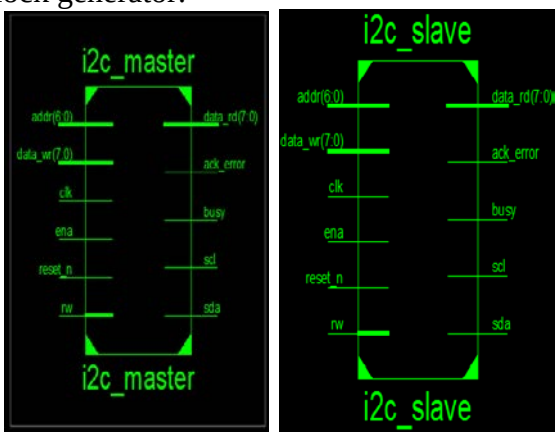


Figure 9: Master /Slave Top module

Table1 indicates the pin and port connection of I^2C Master/Slave.

Port name	width	mode	Description
Clk	1	In	System clock.
reset_n	1	In	Asynchronous active low reset.
Ena	1	In	0: no transaction is initiated. 1: latches in addr, rw, and data_wr to initiate a transaction. If ena is high at the conclusion of a transaction (i.e. when busy goes low) then a new address, read/write command, and data are latched in to continue the transaction.
Addr	7	In	Address of target slave.
Rw	1	In	0: write command. 1: read command.
data_wr	8	In	Data to transmit if rw = 0 (write).
data_rd	8	Out	Data received if rw = 1 (read).
Busy	1	Out	0: I2C master is idle and last read data is available on data_rd. 1: command has been latched in and transaction is in progress.
ack_error	1	Buffer	0: no acknowledge errors. 1: at least one acknowledge error occurred during the transaction. ack_error clears itself at the beginning of each transaction.
Sda	1	In out	Serial data line of I2C bus.
Scl	1	In out	Serial clock line of I2C bus.

Table 1: Pin-port description

B. I^2C Master/Slave state machine

I^2C master and slave state machine explained in Figure10 that is generate all mechanism of I2C-bus protocol. If any address of the device is detected so enable signal goes high and generate the start condition. Before that it will first enter to a ready state. If the start condition is done successfully then it will go to command state. Command state remains as it is if bit count is not equal to 0. It means last bit is not received; then it will communicate the address and rw command towards the bus. If rw bit is 0 then write Operation else read operation. If bit count is 0 (last bit received) of any rw operation than goes to slv_ack1 state that is used to get and verifies the correctness slave's acknowledge. Once read write operation complete, then master will store and verifies the correctness of slave response (slv_ack2 state) for writing & mstr_ack state for reading respectively. Whether enable signal generates another command, then master instantly switch with second write (wr) & read (rd) state whether command (cmd state)is the same as with previous than rw state must be interchangeable. Whether Command state is not same as previous one, If any new slave address is detected then master will generate a repeated start condition(start state) as per the I2C specification. If master finish a read & writes operation also ena signal is not generating any

more command, then the master will issues stop condition (stop state) and go back to the ready state. In slave state machine only state operation of slave acknowledgement and master acknowledgement will change nothing else.

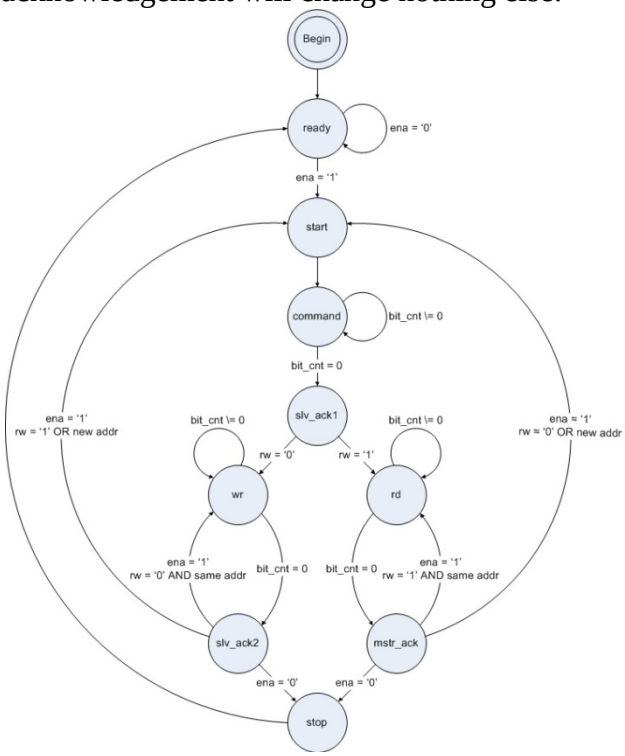


Figure 10: I^{2}C Master Finite state machine

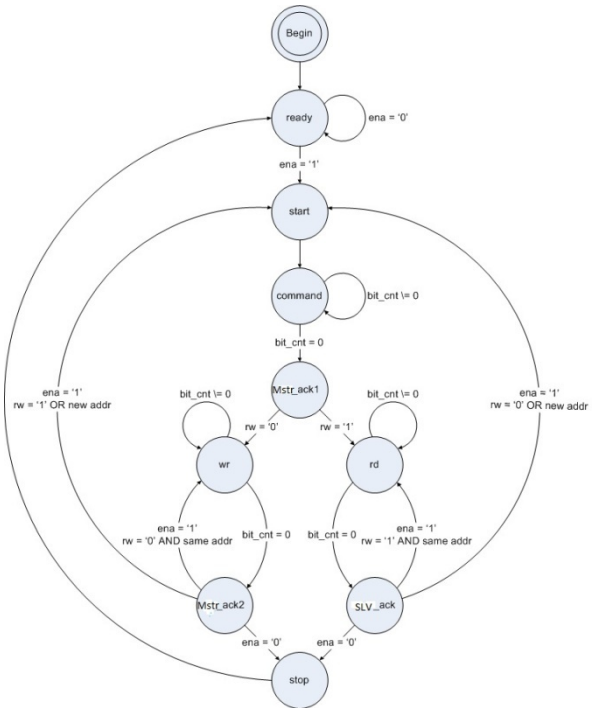


Figure 11: I2C Slave Finite machine State Diagram

IV.IMPLEMENTATION RESULTS

We have done the simulation of complete I2C frame format on Xilinx ISE simulator. Also we cross check the Design work as per the FSM or not. This design is for I^{2}C master to slave communication. The simulation waveform results indicate how data and address transmission are occurring.

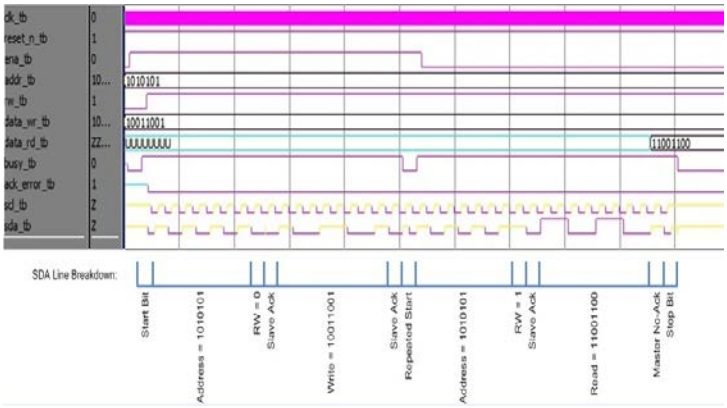


Figure 12: Master to slave Communication.

V.APPLICATIONS

I2C Protocol can be used in many devices like computer camera, EEPROM, Audio-Video recording devices. The example is given below. Only two wires can connect so many devices but at a time one will be activate.[1]

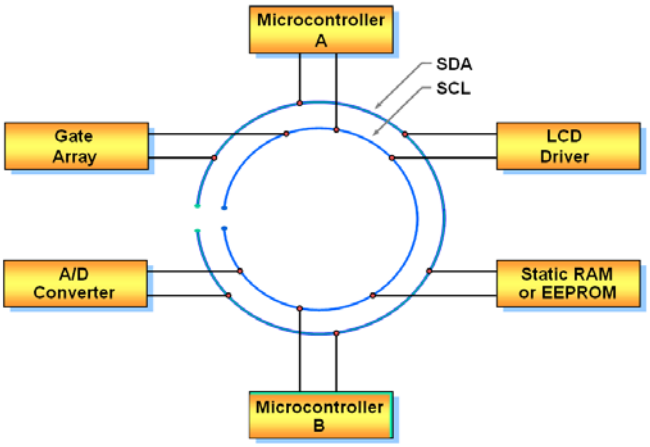


Figure 13: Example of I2C Devices

CONCLUSION

In a nutshell, The I2C master is basically includes programmable logic that can be used to communicate with I2C slaves bypass to a parallel interface. This protocol can be work as a NXP I2C specification only for single master

buses and also added feature of clock stretching.

MOTIVATIONAL WORK

This I2C can be work as a multi master and multi slave. The idea is that we can use multiplexer and de-multiplexer to select particular master and slave selection. We can also verifying test cases using system verilog environment and check all phases in universal verification methodology. Finally check the ASIC backend Flow & Design will be ready for Tap out.

REFERENCES

[1]THE I2C BUS SPECIFICATION VERSION 6.0 4th April, 2014, Philips Semiconductors.

[2]M. Alassir, J. Denoulet, O. Romain & P. Garda: A systemc AMS Model of an I2C Bus Controller.

[3]P.Venkateswaran, Madhumita Mukherjee, Arindam Sanyal, Snehasish Das and R.Nandi: Design and Implementation of FPGA Based Interface Model for Scale-Free Network using I2C Bus Protocol on Quartus II 6.0.

[4]Peter Corcoran: Two wires and 30 years.