ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

Vol. 3, Issue 6, June 2015

# ASIC Implementation of I2CMaster Bus Controller

S Sindhu[1], Ankit KV[2], Dr. Vijaya Prakash A M[3]

M.Tech Student, Dept. of E&C, BIT, Bangalore, KA, India[1]

M.Tech Student, Dept. of E&C, BIT, Bangalore, KA, India[2]

Professor, Dept. of E&C, BIT, Bangalore, KA, India[3]

**ABSTRACT:**ASIC Implementation of I2C Master bus controller has been proposed in this paper. I2C is one the most dominant protocol for on chip communication between different modules. The FPGA implementation of I2C master controller contains many features to incorporate vast varieties of applications and I2C standard; hence it's bulky, slow with high power requirement. While, all features of generic design are rarely used fully in any particular IC. Hence, a modified design with fewer features but withhigh power efficiency, less delay and less area requirement, has been proposed in this paper. This design is best suitable for any ASIC design where for on-chip communication, a serial bus interface is required. Moreover, Proposed I2C Master Controller has been designed for ASIC, which makes the design highly portable on any SOC chip as well. The entire custom ASIC implementation of proposed design has been done in Cadence Tool chain with 45nm technology standard library. A thorough comparison has been done between FPGA implementation of I2C Controller and the ASIC implementation of the proposed design.

**KEYWORDS**:I2C Master controller; Serial protocol; ASIC implementation; Custom ASIC design;Serial communication.

## I. INTRODUCTION

Protocols are implemented on ICs by communication buses controlled by master controller modules. Communication Protocols generally divided into two broad categories: Parallel and Serial. Parallel buses for all the interfaces are not a good trade-off between cost, time, power and performance. Alternate serial buses are much efficient in on-board data communication between different sub-systems on an IC or SOC. Most of the peripherals on modern ASIC & SOC designs use serial communication buses for data transfers between processor or between processor and peripherals.

The objective of this paper is to develop a concise module of I2C Master Bus Controller for ASIC and other applications like SOCs, wherever there is a requirement of compact & small customized I2C Protocol Master Bus controller module for communication between on-board components. Now, the VLSI design industry evolving towards smaller sizing. To cope up with this trend, all the inter sub modules of a IC whether main Logical blocks or intercommunication protocols modules like I2C, all have to evolved into new size scaled down versions from bulky past versions [2,7].

Hence an ASIC implementation of I2C bus module will bring low running cost, high performance, power efficiency and conciseness to the already available versions, which are generally made for FPGAs.

This ASIC implementation of master controller will have fewer features but very small size, min. delay and low power requirements due to modifications in design for specific applications and also due to technology mapping of 45nm technology library, which is the recent outbreak in VLSI technology.

DOI: 10.15680/ijircce.2015.0306112

## II. RELATED WORK

Few examples of Serial communication protocols are UART, CAN, USB, SPI and Inter IC [2, 4]. Most of them are the one point to point type protocol [4]. While growing trend in USB plug & play devices making USB protocol famous for external device communication but main drawback of USB is of using multiplexer to communicate with other devices make it bad option for on chip communication protocol.On the other hand other protocols like I2C and CAN protocol uses software addressing. This feature allows bringing more number of slaves and masters in the communication, hence becoming a good choice for on-chip communication. Now out of these two protocols only I2C can be designed in a simple way and it's flexible to maintain [5, 6]. Only I2C and CAN protocol uses software addressing but only I2C is convenient to design for ICs because CAN is specifically designed for automobiles [10].

- **UART:** It has Limited Functionality, seed limit & it is one point to one point [7].
- **CAN:** It has very Limited Portfolio & it is Automotive oriented [4].
- **USB:**It Require a powerful master host controller and extra drivers required for plug-in.
- **SPI:**This protocol don't have plug and play hardware feature plus there is no fixed standard for this and there is no provision for acknowledgement [9].
- **I2C:**This protocol is simple plug & play and cost effective**.** Moreover it follows one universally accepted standard [1].

Compared with the parallel bus, less wiring is required in I2C, fewer connection pins and as a result a less number of traces required on PCB boards. I2C have many applications in IC design [10]. One of the applications of I2C is data surveillance, to get a better accuracy and efficiency. In data surveillance system, the I2C protocol is designed in such a way that, it is easy to communicate without any data loss between different devices, and also compared with the other the communication protocols, an excellent speed is developed [6, 8].

## III. I2C (INTER-INTEGRATED CIRCUIT) PROTOCOL

I²C protocol with pronunciation as I-squared-C has features like multi slave/master support, serial communication and single end control. NXP semiconductors now Philips Semiconductor, had invent I2C protocol [12] for attaching low-speed peripherals with computer main motherboards.

I²C uses only two bidirectional open-drain lines, Serial Data Line (SDA) and Serial Clock Line (SCL), pulled up with resistors [3]. Typical voltages used are +5 V or +3.3 V although systems with other voltages are permitted. Standard I2C devices operate up to 100Kbps, while most of the fast mode I2C devices available today support up to 3.4Mbps operation [12].

### A. *I2C Specifications:*

I²C bus common speeds are up to 10 kbps in low-speed mode and arbitrarily little low clock frequencies are also allowed sometimes [1]. Recent versions of I²C protocol can host more number of nodes There is also one other new features of 16-bit addressing. Actual user data transfer rate is lower than peak bit rates in general. Like if each interaction with a slave inefficiently allows only 2 byte of data transfer then the data rate will be less than half the peak bit rate. The design mentioned in figure1 shows bus with a clock (SCL) and data (SDA) lines have 7bit addressing mode. The bus can take any of two roles for nodes, a master or a slave:

- Master node — this node will generates clock and initialize the communication with slaves
- Slave node — these nodes will receives clock and responds when addressed by master.

The bus controller is a multi-master bus hence it can support many number of master nodes. Moreover, the master and slave roles will be changed between messages (after a STOP sequence is sent). For a given bus device there may be four potential modes of operation but most devices use only single role and their two modes where either slave or master sends data, one at a time.

DOI: 10.15680/ijircce.2015.0306112

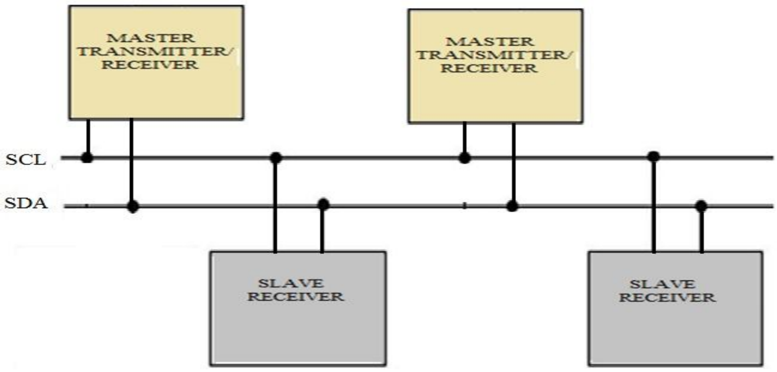


Fig.1. I2C bus configuration using masters and slaves

B. *Working of I2C:*

Initially the master will come in master transmit mode and start sending a start bit and after that the 7-bit address of the slave device it wishes to communicate which is followed by a single bit (the 8th bit) representing whether it wishes to write(0) to or read(1) from the slave device. If any slave with the sent address exists in the bus line then it will respond with an ACK bit [1] (active low for acknowledged) for that address sequence. The master then stays in either transmit or receive mode (as per the read/write bit it sent) while slave continues with the complementary mode (receive or transmit, respectively). The address and the data bytes are sent as MSB (most significant bit) first. Start bit is indicated by the high-to-low transition of SDA while the SCL kept high. And the stop bit is indicated by low-to-high transition on the SDA line keeping SCL high and rest of the transitions of SDA line take place with SCL low [3,7,8].

## IV. Design Methodology

The following section contains all the integral information about the designing of I2C Master Controller with all components and their description with block diagrams.

A. *I2C Master Controller:*

It is the top module which encapsulates a FIFO and a FSM. It is the top module which allows a proper communication between the sub modules such as the FIFO and the FSM. The FIFO and the FSM are sequential and are synchronous in operation with each other, which means both the sub-modules works with the same clock frequency, which is generated by the testbench of the i2c_master_controller. [11]

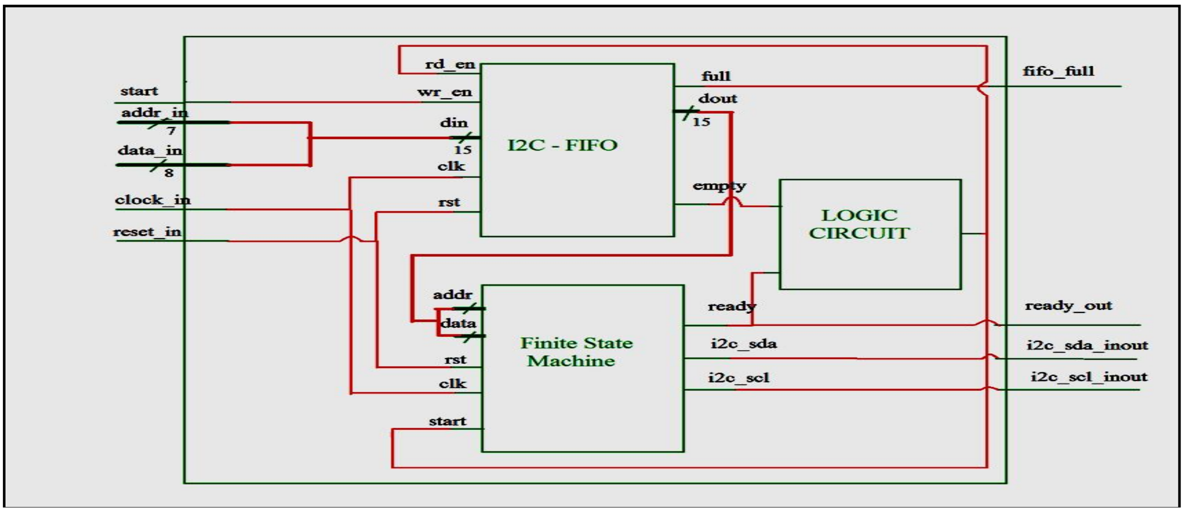


Fig. 2: Block Diagram of I2C Master Controller

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

Vol. 3, Issue 6, June 2015

B. *I2C-FIFO:*

It is a deep data structure which implements the simple “First In First Out” architecture. The width of the FIFO is of 15 bits for this proposed design. Whenever the FIFO is full, its filled status is indicated by the status output signal “full” and whenever there are no values stored in the FIFO, its empty status is indicated by the status signal “empty”.

C. *Finite State Machine:*

FSM is the main component of the I2C master controller in this project.The FSM defined as sequential circuit which uses the finite number of states to track the history of operations. Eight states are present in the FSM to get the actual working of the I2C.

In general I2C master controller components consists of a Finite State Machine, Clock generator, Start/Stop controller and counter. In this design, to reduce area penalty, instead of using individual modules for clock generator, start/stop controller, counter and FSM (Finite State Machine), the logic for the FSM is written in such a way that it performs the functionality of a counter, clock generator, start/stop controller inside only.

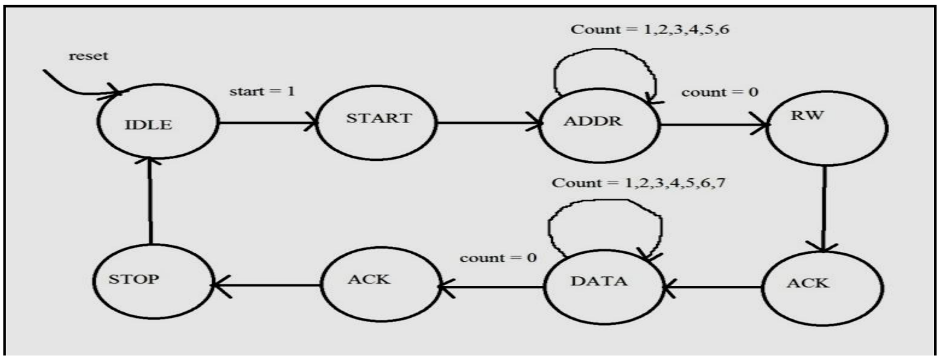


Fig 3: Proposed Finite State Machine Diagram

D. *I2C Clock Generator:*

It generates the I2C clock signal, which works at a frequency of 100 kbps (standard mode). To generate 100kbps frequency clock, the board clock signal has to be generated by 512 times in the clock divided module. The Start, stop and idle states in the FSM only requires the SCL to be high. Rest of the operations of the SDA line requires negative edge of the SCL clock. This module is used inside the testbench for functional verification [11].

## V. IMPLEMENTATION OF DESIGN

The I2C-Master-Controller has been divided into two modules:

- I2C-FIFO
- Finite State Machine

I2C FIFO and the Finite State Machine has been designed using Verilog HDL. FSM is a sequential circuit that uses a finite number of states to keep track of its history of operations, and the next state is determined based on history of operation and current input. There are several states in obtaining the functionality of I2C Master Controller operation.

A. *I2C Master Controller:*

The Figure 5 shows module of the I2C-master-controller with all the input and output signals.

DOI: 10.15680/ijircce.2015.0306112

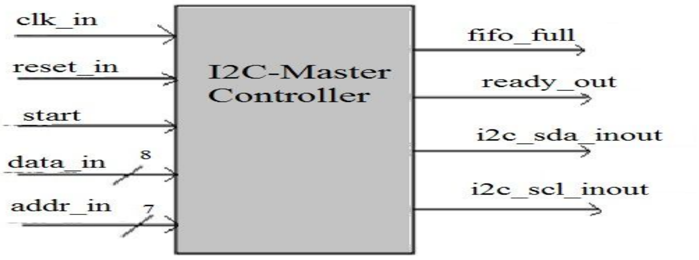


Fig.5: I2C-Master-Controller Module Block Diagram

a) *Algorithm for I2C Master Controller:*

Step 1: Module I2C Master Controller
Step 2: Define Inputs: clk_in, reset_in, start, addr_in, data_in. and Inouts:- I2C-sda-inout, I2C-scl-inout.
Step 3: Define Output - FIFO-full, Ready_out.
Step 4: Instantiate I2C-FIFO module.
Step5: Instantiate Finite State Machine module.

B. *I2C FIFO:*

The Figure 6 shows module of the I2C-FIFO with all the input and output signals.

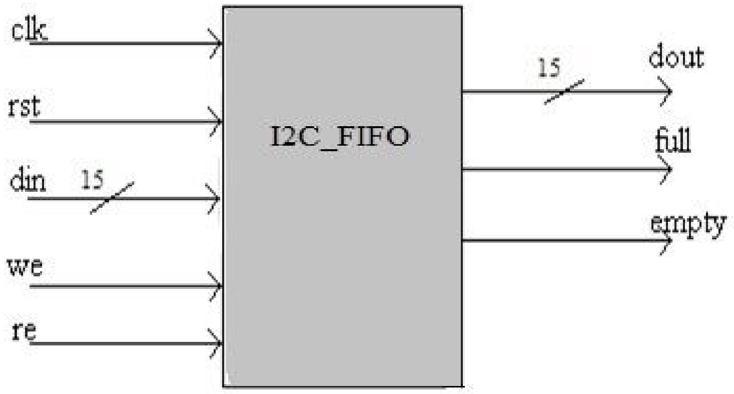


Fig.6: I2C-FIFO Module

*a) Algorithm for I2C-FIFO:*

Step1: Module I2C_FIFO.
Step2: Define Input – clk, rst, din, we, re.
Step3: Define Output – dout, full, empty.
Step4: If clk and rst are equal to 1, then initialize the wptr and rptr to zero.
Step5:If reset is equal to zero and we is high, then increment the write pointer by 1 while If re is high, then increment the read pointer by 1.
Step6: If wptr is equal to rptr , then empty signal goes high. And reading is not possible from the FIFO
Step7: If wptr + 1 is equal to rptr, then full signal goes high and Writing is not possible in to the FIFO

C. *Finite State Machine:*

*a) Algorithm for Finite State Machine:*

Step1: An idle State: I$^2$C bus will be in idle state. (SCL and SDA remains logical high).
Step2: Start condition: If the start bit given high, and the sda line is made low when scl line is high.
Step3: Slave address - write: master sends the slave address to write to the slave.0

Step4: If the address matches with the slave address, it sends an acknowledgement by pulling the sda line low.
Step5: Master sends the 8-bit data on the sda line. After receiving the data, slave acknowledges by pulling the sda line low.
Step6: Stop condition: After the transmission, master sends the STOP sequence by pulling the sda line high when the scl clock is high.
Step7: Master transmits slave address and the read signal for read operation to the slave.
Step8: After master receives data from the slave it sends acknowledgement in return to slave.
Step9: Master sends a STOP bit to terminate the connection (SCL is high and SDA is from Low to high).

The Figure 7 shows the flowchart of the steps that the I2C-Master-Controller is following to complete the process of transmission from master to the slave.

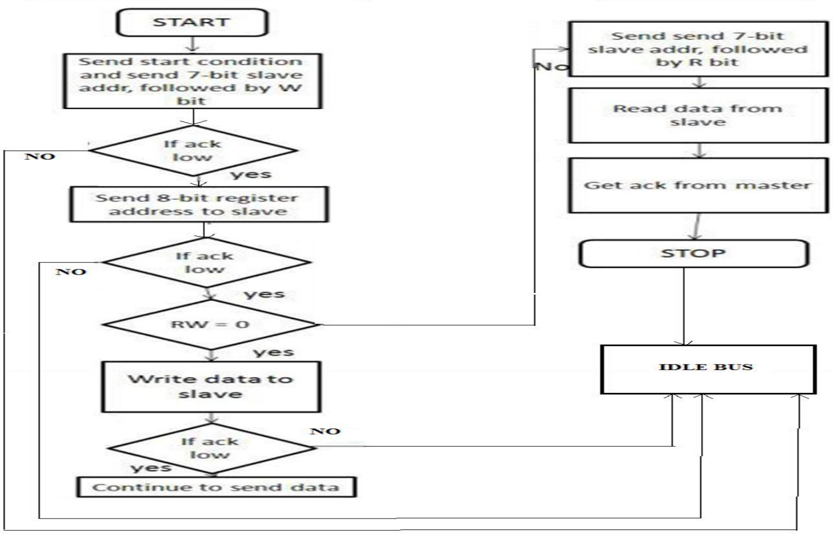
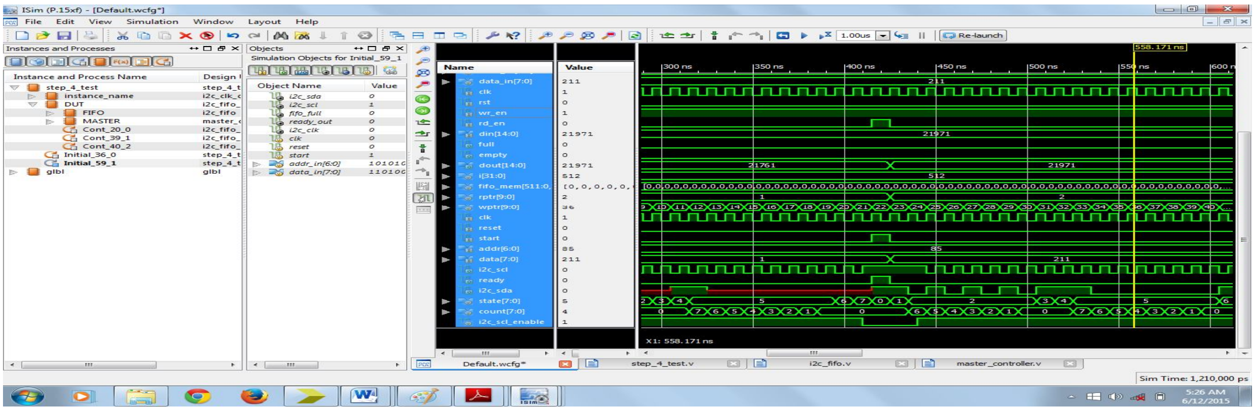


Fig-7: Flowchart for Finite State Machine

## VI. Results And Discussion

### A. *Simulation Results:*

Fig.8 shows the Transmission of data and address in the master controller by storing the data and address in the FIFO and the driving the values to the FSM.



| Name | Value |
|---|---|
| data_in[7:0] | 211 |
| clk | 1 |
| rst | 0 |
| wr_en | 1 |
| rd_en | 0 |
| din[14:0] | 21971 |
| full | 0 |
| empty | 0 |
| dout[14:0] | 21971 |
| i[31:0] | 512 |
| fifo_mem[511:0, | [0,0,0,0,0, |
| rptr[9:0] | 2 |
| wptr[9:0] | 36 |
| clk | 1 |
| reset | 0 |
| start | 0 |
| addr[6:0] | 85 |
| data[7:0] | 211 |
| i2c_scl | 0 |
| ready | 0 |
| i2c_sda | 0 |
| state[7:0] | 5 |
| count[7:0] | 4 |
| i2c_scl_enable | 1 |

Fig.8: Simulation Output of I2C_Master_Controller

DOI: 10.15680/ijircce.2015.0306112

B. *Synthesis Results:*

The inner view of the RTL schematics of I2C_Master_Controller is shown in the figure 10.

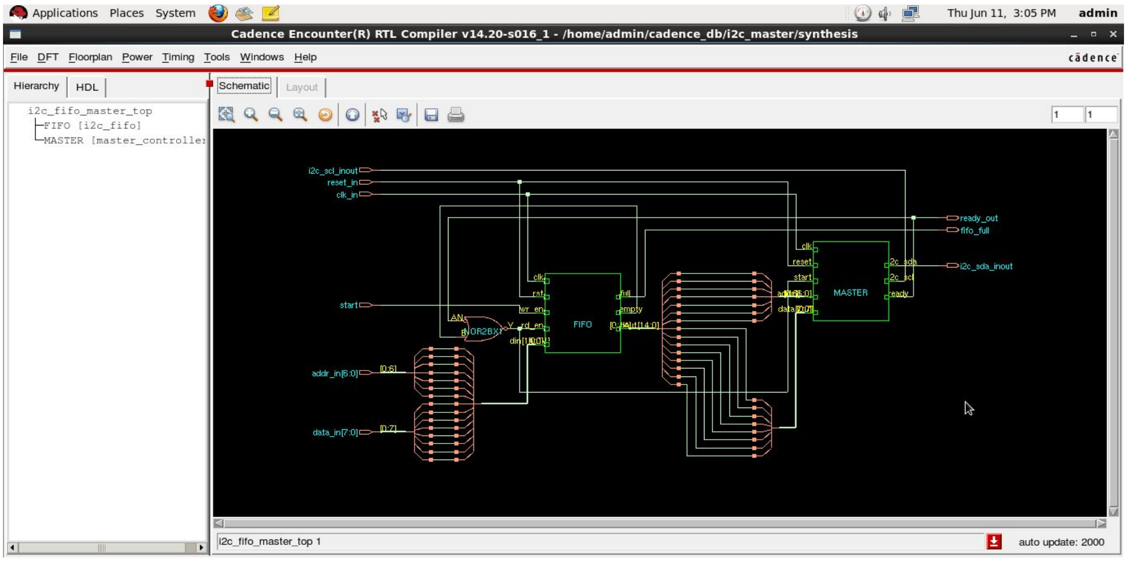
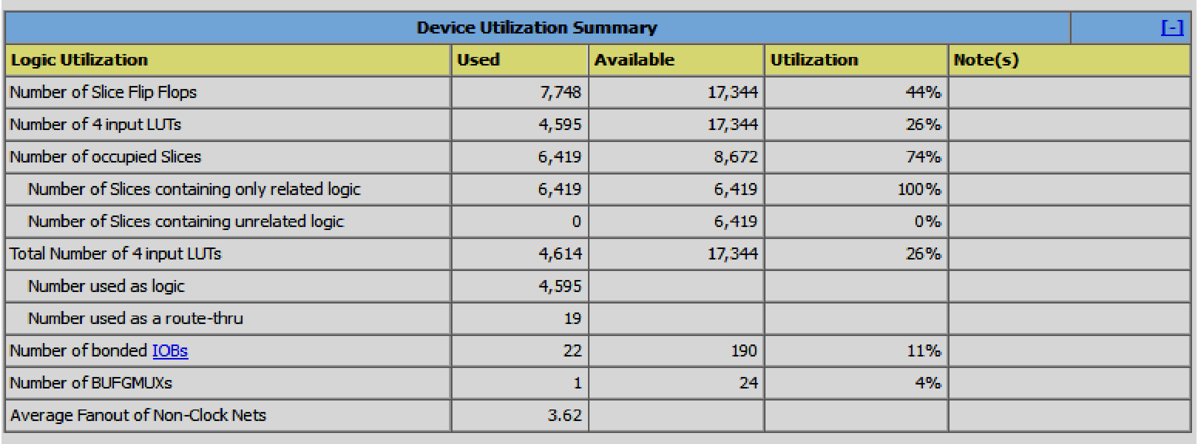

Applications Places System
Thu Jun 11, 3:05 PM admin
Cadence Encounter(R) RTL Compiler v14.20-s016_1 - /home/admin/cadence_db/i2c_master/synthesis
File DFT Floorplan Power Timing Tools Windows Help
cadence
Hierarchy HDL
Schematic Layout
i2c_fifo_master_top
FIFO [i2c_fifo]
MASTER [master_controller
i2c_scl_inout
reset_in
clk_in
start
addr_in[6:0]
data_in[7:0]
ready_out
fifo_full
i2c_sda_inout
FIFO
MASTER
OR2BX
i2c_fifo_master_top 1
auto update: 2000

Fig.9: RTL schematic of I2C-Master-Controller of Proposed Design

*a) FPGA Design's Synthesis Reports:*

The advanced HDL synthesis report is generated by Xilinx ISE, indicating realized macro statistics. The implementation results are tabulated in below table 1.

Table 1: Device Utilization Summary of FPGA Implementation



| Device Utilization Summary | | | | [-] |
|---|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** |
| Number of Slice Flip Flops | 7,748 | 17,344 | 44% | |
| Number of 4 input LUTs | 4,595 | 17,344 | 26% | |
| Number of occupied Slices | 6,419 | 8,672 | 74% | |
| Number of Slices containing only related logic | 6,419 | 6,419 | 100% | |
| Number of Slices containing unrelated logic | 0 | 6,419 | 0% | |
| Total Number of 4 input LUTs | 4,614 | 17,344 | 26% | |
| Number used as logic | 4,595 | | | |
| Number used as a route-thru | 19 | | | |
| Number of bonded IOBs | 22 | 190 | 11% | |
| Number of BUFGMUXs | 1 | 24 | 4% | |
| Average Fanout of Non-Clock Nets | 3.62 | | | |

Table2 gives the timing report of the I2C Master Controller for the FPGA Design Flow,which is implemented using XST in Xilinx.

Table 2: Timing Report of FPGA Implementation

| | |
|---|---|
| Minimum period<br>(Maximum Frequency: 115.640MHz) | 8.648ns |
| Minimum input arrival time before clock | 7.423ns |
| Maximum output required time after clock | 8.171ns |

Table3 gives the power report of the I2C Master Controller for the FPGA Design Flow,which is implemented using XST in Xilinx.

Table 3: Power Report of FPGA Implementation

On-Chip Power Summary

| On-Chip | Power (mW) | Used | Available | Utilization (%) |
|---|---|---|---|---|
| Clocks | 15.45 | 1 | --- | --- |
| Logic | 0.11 | 4613 | 17344 | 27 |
| Signals | 15.44 | 12076 | --- | --- |
| IOs | 9.43 | 22 | 190 | 12 |
| Quiescent | 160.03 | | | |
| Total | 200.47 | | | |

Power Supply Summary

| | Total | Dynamic | Quiescent |
|---|---|---|---|
| Supply Power (mW) | 200.47 | 40.44 | 160.03 |

*b) ASIC Implementation:*

Table 4 gives the timing report of the I2C Master Controller for the ASIC Design Flow, which is implemented using RC compiler in Cadence.

Table 4:Timing Report of ASICImplementation

| | |
|---|---|
| Cost Group | 'clk_in' (path_group 'clk_in') |
| Timing slack | 3849ps |
| Start-point | reset_in |
| End-point | MASTER/i2c_scl_enable_reg/D |

Total Delay = Total time given- slack = 5000ps - 3847ps = 1153 ps

Table 5 gives the power report of the I2C Master Controller for the ASIC Design Flow, which is implemented using RC compiler in Cadence.

Table 5:Power Report of ASICImplementation

| Instance | Cells | Leakage Power(nW) | Dynamic Power(nW) | Total Power(nW) |
|---|---|---|---|---|
| i2c_master_controller | 26888 | 1589.132 | 2724365.020 | 2725954.152 |
| I2C-FIFO | 26774 | 1583.283 | 2577280.220 | 2578863.503 |
| FSM | 113 | 5.821 | 14997.857 | 15003.679 |

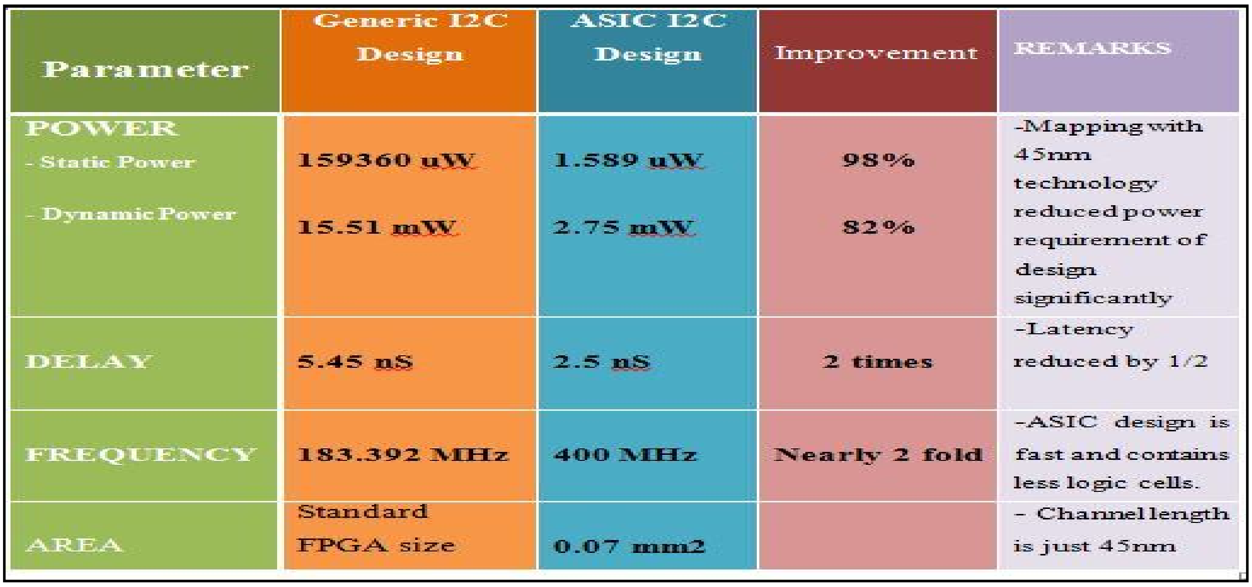
## VII. Conclusion And Future Scope

The new modified I2C master controller designed in this paper is much efficient than any existing I2C controller till date. To check and analyse the improvements done in this design, a thorough comparison has been done between FPGA implementation and proposed ASIC version of master controller bus. First, thorough simulation and elaboration has been done on Xilinx Tool to get the attributes of the FPGAimplementation which includes large number of blocks and logic implementation.

Next, the whole design has been modified for ASIC implementation and number of inter blocks has been reduced by significant amount using FSM, which leads to a highly optimized and efficient outcome.
Improvement has been observed in these Performance parameters: Frequency of Operation, Delay, and Power Consumption.

Table 7 gives the comparision Comparison Summary between FPGA implementation of I2C Master Controller RTL design and ASIC implemented I2C Master Controller module.

Table6. FPGA Implementation vs. ASIC Implementation for I2C Master Controller



| Parameter | Generic I2C Design | ASIC I2C Design | Improvement | REMARKS |
|---|---|---|---|---|
| POWER<br>- Static Power<br>- Dynamic Power | 159360 uW<br>15.51 mW | 1.589 uW<br>2.75 mW | 98%<br>82% | -Mapping with 45nm technology reduced power requirement of design significantly |
| DELAY | 5.45 nS | 2.5 nS | 2 times | -Latency reduced by 1/2 |
| FREQUENCY | 183.392 MHz | 400 MHz | Nearly 2 fold | -ASIC design is fast and contains less logic cells. |
| AREA | Standard FPGA size | 0.07 mm2 | | - Channel length is just 45nm |

## REFERENCES

[1] ArvindSahu, Dr. Ravi Shankar Mishra, PuranGour “An Implementation of I2C using VHDL for DATA surveillance”, International Journal on Computer Science and Engineering (IJCSE) ,Vol. 3, 2011.
[2] M.Jyothi, L.Ravi Chandra, M.Sahithi, S.DayaSagarChowdary, K.Rajasekhar, K.Purnima, “Implementation of SPI Communication Protocol for Multipurpose Applications with I2C Power and Area Reduction”, International Journal of Engineering Research and Applications (IJERA) , Vol. 2, Issue 2, pp.875-883,2012.
[3] BollamEswari, N.Ponmagal, K.Preethi, S.G.Sreejeesh “Implementation of I2C Master Bus Controller on FPGA”, International conference on Communication and Signal Processing,pp,1113-1116, 2013.
[4] A.K. Oudjida, M.L. Berrandjia, R. Tiar, A. Liacha, K. Tahraoui “FPGA Implementation of I2C & SPI Protocols: A Comparative Study” –IEEE Conference on Electronics, Circuits and Systems, pp. 507 - 510,2009.
[5] Fang Zhao ; Daquan Deng ; Zhensheng Wang ; Huanzhao Liu “Design of Schematic Mapping System Based on I2C and USB Bus” – 2011 First International Conference on Instrumentation, Measurement, Computer, Communication and Control, pp. 180-183,2011.
[6] Oberg, J.; Wei Hu ;Irturk, A. ; Tiwari, M. “Information flow isolation in I2C and USB” –International Conference on Design Automatic Conference(DAC),pp.254-259, 2011.
[7] Narendra Gupta, Vipin Gupta “The Performance and Optimization of I2C Protocol using Verilog on Xilinx Tool” International Journal Of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering, vol.1, issue 9, 2013.
[8] MR. 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,Volume.02, Issue .02,2013.
[9]Nam Nguyen “Design of I2C Interface for Custom ASICS Used in the Detection of Ionizing Radiation” Southern Illinois University Edwardsville Edwardsville, Illinois,2008.
[10] Tatiana Leal-del Río, Gustavo Juarez-Gracia, L. NoéOliva-Moreno “Implementation of the communication protocols SPI and I2C using a FPGA by the HDL-Verilog language”, Research in Computing Science, pp. 31–41, 2014.

## BIOGRAPHY

**Dr. VijayaPrakash AM** is a Professor in Dept. of E&C, BIT, Bangalore, KA, India. He has received Ph.D. in VLSI. His research interests are Low Power VLSI, Synthesis and Optimization. He is a member of IMAPS and ISTE.

**S Sindhu**is a PG scholar in Dept. of E&C, BIT, Bangalore, KA, India. She received her B.E degree in 2012 from JNTU, Andhra Pradesh, India. Her research interests are VLSI verification and communication protocols etc.

**Ankit KV** is a PG Scholar in Dept. of E&C, BIT, Bangalore, KA, India. He received his B.Tech (Hons.) degree in 2012 and now involved in active research onVLSI Mixed Mode Designs, Logic Designing & Electronics Products etc.

DOI: 10.15680/ijircce.2015.0306112