ASIC IMPLEMENTATION OF I^2C MASTER BUS CONTROLLER FIRM IP CORE

S Sindhu, Dr. Vijaya Prakash A M and Ankit K V

Department of E & C, Bangalore Institute of Technology, Bangalore, KA

ABSTRACT

ASIC Implementation of I^2C Master bus controller with design of Firm IP core has been proposed in this paper. I^2C is one the most prominent protocol used in on chip communication among sub-systems. The generic design of I^2C master controller has ample of features to incorporate vast varieties of application and I^2C standards. The generic design is slow, congested and require high power. It's rare to utilize all the features of generic design fully in a single particular application or system. Hence, a modified ASIC design with specific less features but with better timing, low power requirement and less area overhead, has been proposed in this paper. This design is specifically apt for digital systems which have serial bus interface requirement for on board communication. Moreover, the Firm IP core of I^2C Master Controller has been designed for ASIC, which makes the design highly portable on any ASIC chips or SOC designs. The firm IPs is best in terms of flexibility and more predictable than commonly found soft IPs. The entire custom ASIC implementation of proposed design has been done in Cadence Tool chain with 45nm technology using standard cell library. A thorough comparison has been done between generic open sourced RTL design of I2C Controller obtained from Opencores.org and our proposed design.

KEYWORDS

Serial bus interfaces, I^2C *Protocols, ASIC designing, Firm IP Core, on-chip communications, IP designing.*

1. INTRODUCTION

Communication protocols for data interchange are usually incorporated on digital systems using an on-chip master controller sub-system. The protocols used for communication are 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.

Few examples of Serial communication protocols are UART, CAN, USB, SPI and Inter IC [2, 4]. The USB, SPI and UARTS can communicate form one device to only one other at a time, i.e. not more than two devices can be connected with each other at a time. Whereas USB has problem of using multiplexer for communication with other devices which makes it bulky [5, 6] . Only I^2C and CAN protocol uses software addressing but only I^2C is convenient to design for ICs because CAN is specifically designed for automobiles.

The objective of this paper is to develop a Firm IP [9] core of I^2C Master Bus Controller for ASIC and other applications like SOCs, wherever there is a requirement of compact & small customized I^2C 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

DOI : 10.5121/vlsic.2015.6405

modules like I^2C, all have to evolved into new size scaled down versions from bulky past versions [2,7].

Hence an ASIC implementation of I^2C bus module will bring low running cost, high performance, power efficiency and conciseness to the already available versions, which are generally made for FPGAs. Moreover Firm IP Core is a recent revolution in IP development industry. So in this paper a firm IP core design of I^2C master controller bus has been proposed which has synthesized netlist & area, power, timing information as per 45nm standard technology library.

This ASIC implementation of master as firm IP core will have few 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.

2. I^2C (INTER-INTEGRATED CIRCUIT) PROTOCOL

I^2C 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 I^2C protocol [9] for attaching low-speed peripherals with computer main motherboards.

I^2C 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 I^2C devices operate up to 100Kbps, while most of the fast mode I^2C devices available today support up to 3.4 Mbps operation [8]. One intermediate mode is also available which usually works in range of 400Kbps.

2.1. I^2C Specification

Standard I^2C 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 I^2C 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.

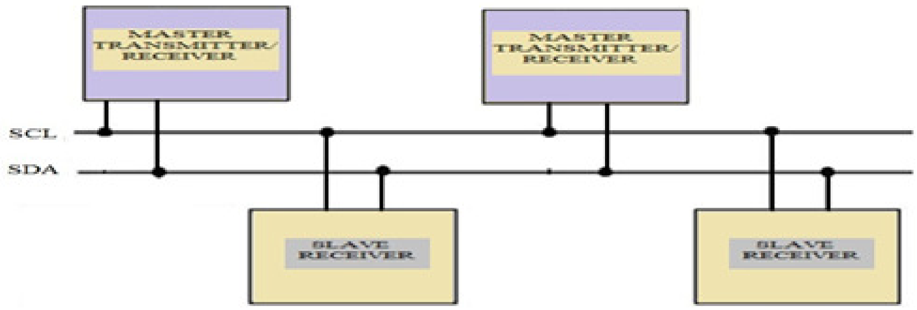

Figure 1. I^{2}C bus configuration using Master and Slave

2.2. Working of I^{2}C

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].

3. DESIGN METHODOLOGY

The following section contains all the integral information about the designing of I^{2}C Master Controller with all components and their description with block diagrams.

3.1. I^{2}C Master Controller

It is the top module which encapsulates a FIFO and a FSM. It 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.

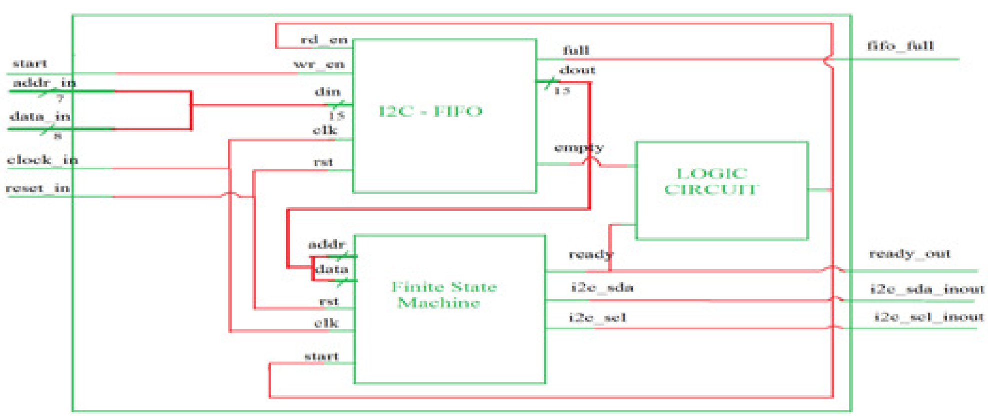

Figure 2. Block Diagram of I^{2}C Master Controller

3.2. I^2C-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".

3.3. Finite State Machine

FSM is the main component of the I^2C 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 I^2C[12].

In general I^2C 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, and start/stop controller inside only.

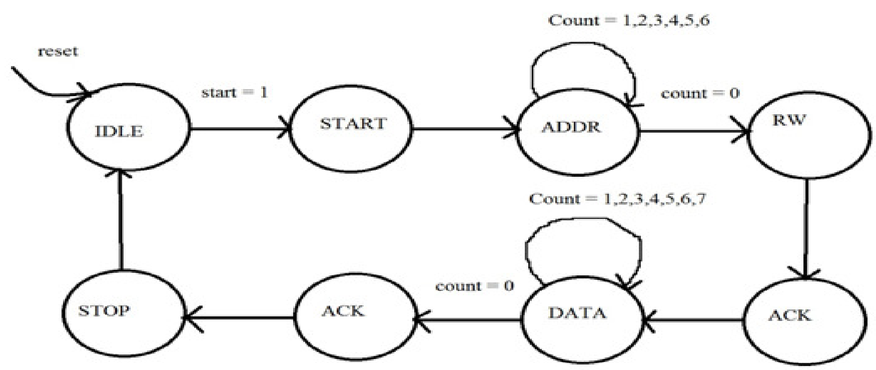

Figure 3. Proposed Finite State Machine Diagram

3.4. I^2C Clock Generator

It generates the I^2C 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.

3.5 Counter

In the FSM design described above, a local variable is declared which acts like a counter. This counter will keep track of both the address and data. It Reduce the need of two counter blocks respectively for address and data, thus reducing area penalty. The 3-bit local variable, which acts like a counter counts the 7-bit address and the 6-bit data.

3.6 Start/Stop Controller

The Start/Stop Control signals generate and detect start and stop events on the I2C bus. The start and stop condition detection is necessary to determine whether I2C bus used by another master on the bus when there are multiple masters on same I2C line. Furthermore, the start detection is necessary for the primary I2C Master Controller because it cannot proceed with a transaction until the start condition has been accepted by the I2C bus.

3.7 Intellectual Property (IP)

IP's are the pre-developed integration of pre-developed pieces of functionality in the IC or SOC design. These IP blocks help the designer to reduce the development cycle time significantly. But, when an attempt is made to integrate multiple IP's, the SOC or IC designer has faced tremendous challenges in understanding these pre-defined functional IP's and the challenges to made the IP's to synchronize with the rest of the IC/SOC. So, in order to avoid the complexity, the IP component developers and the SOC houses has defined standards for design and integration of reusable IP's. The three main standards they have come up with are:

3.7.1 Hard IP

It consists of hard layouts using particular design libraries and is delivered in masked-level designed blocks. These cores give optimized design and the highest performance for the particular physical library. The main disadvantage of hard cores is they are technology dependent and also they provide minimum flexibility and portability [9,11].

3.7.2 Soft IP

Soft IP cores are delivered as RTL VHDL/Verilog code to provide functional descriptions of IP's and provides maximum flexibility and re-configurability. But disadvantage is they must be synthesized, optimized before integration [9, 11].

3.7.3 Firm IP

It has the advantages of both the hard IP and the soft IP. It balances the high performance and optimization properties of the hard IP's with the flexibility of soft IP's. These cores are delivered in the form of targeted netlist for the specific physical libraries after performing synthesis without performing the physical layout. The designers can optimize cores for their specific design needs as the Firm IP blocks are have parameterized circuit descriptions. As the parameters are flexible, it allows the designers to make the performance more predictable [9, 11].

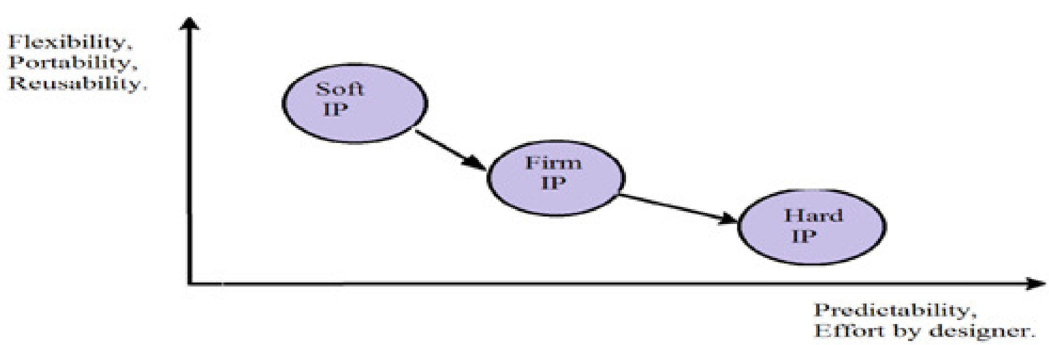

Figure 4. Comparison of different IP cores

4. IMPLEMENTATION OF DESIGNESIGN METHODOLOGY

The I^2C -Master-Controller has been divided into two modules:

- I^2C -FIFO
- Finite State Machine

I^2C FIFO and the Finite State Machine [12] have been designed using Verilog HDL. FSM is a sequential circuit that uses a finite number [12] 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 I^2C Master Controller operation.

4.1. I^2C Master Controller

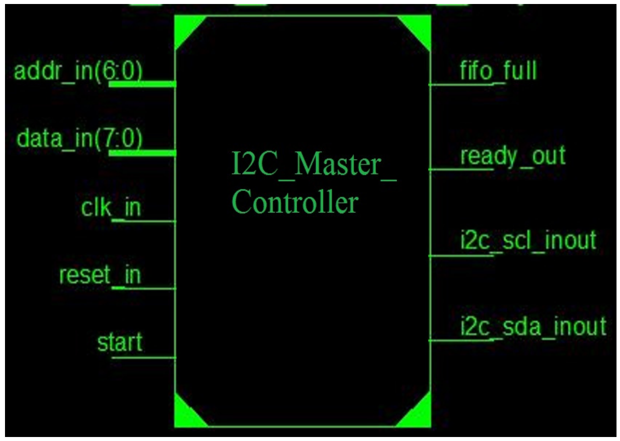

Figure 5. I^2C -Master-Controller Module Block Diagram

4.1.1. Algorithm for I^2C Master Controller:

Step 1: Module I^2C Master Controller

Step 2: Define Inputs: clk_in, reset_in, start, addr_in, data_in. and Inouts:- I^2C -sda-inout, I^2C -scl-inout.

Step 3: Define Output - FIFO-full, Ready_out.

Step 4: Instantiate I^2C -FIFO module.

Step 5: Instantiate Finite State Machine module.

4.2. I^2C FIFO

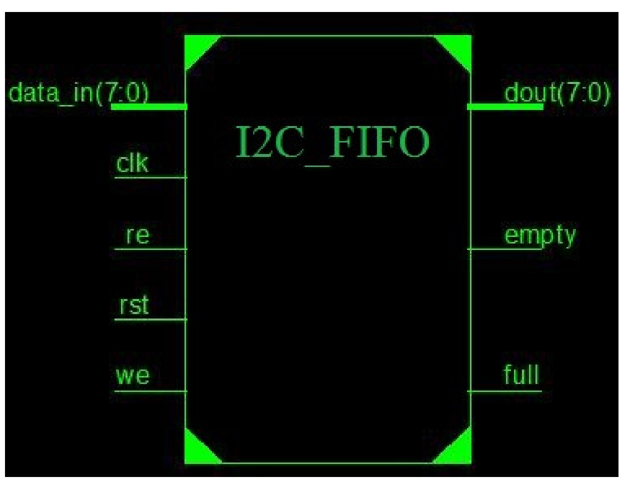

Figure 6. I^2C-FIFO Module.

4.2.1. Algorithm for I^2C-FIFO

Step1: Module I^2C_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.

4.3. Finite State Machine

Algorithm for Finite State Machine:
Step1: An idle State: I^2C 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.

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).

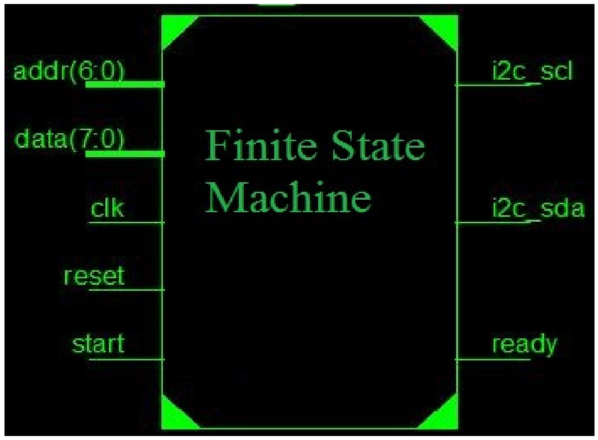

Figure 7. Finite State Machine Module

5. RESULTS AND DISCUSSION

5.1. 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.

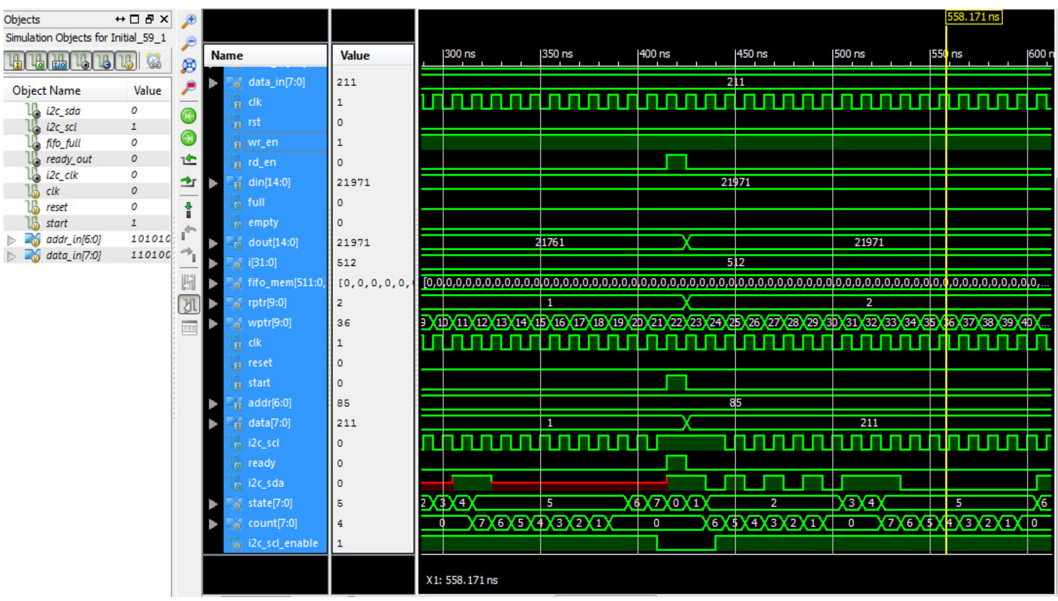

Figure 8. Simulation waveform of I²C_Master_Controller System

5.2. Synthesis Results

The inner view of the RTL schematics of I²C_Master_Controller is shown in the figure 9.

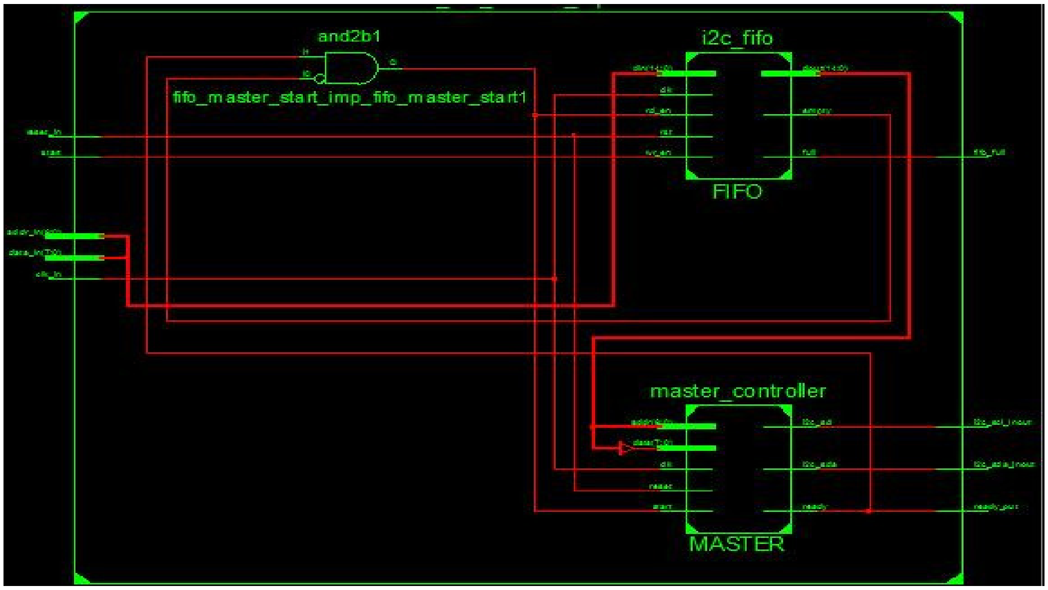

Figure 9. RTL schematic of I^{2}C-Master-Controller of Proposed Design

5.2.1 Generic 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 Generic Design
(Open cores RTL code of I2C master bus controller)

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			

Table 2: Timing Report of Generic open course I2C Design

Minimum period (Maximum Frequency: 183.392MHz)	5.453ns
Minimum input arrival time before clock	5.983ns
Maximum output required time after clock	4.134ns

Table 3: Power Report of Generic open course I2C Design

On-Chip Power Summary

On-Chip	Power (mW)	Used	Available	Utilization (%)
Clocks	5.06	1	---	---
Logic	0.53	226	17344	1
Signals	1.22	279	---	---
IOs	8.70	33	190	17
Quiescent	159.36			
Total	174.87			

Power Supply Summary

	Total	Dynamic	Quiescent
Supply Power (mW)	174.87	15.51	159.36

5.2.2 Proposed ASIC Design's Synthesis Reports

Table 4: Timing Report of proposed ASIC Implementation

Cost Group	'clk_in' (path_group 'clk_in')
Timing slack	1ps
Start-point	reset_in
End-point	MASTER/i2c_scl_enable_reg/D

Total Delay = Total time given- slack = 2501ps - 1ps = 2500 ps

Table 5: Power Report of proposed ASIC Implementation

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

Table 6: Area Report of proposed ASIC Implementation

Instance	Cells	Cell Area	Net Area	Total Area
i2c_master_controller	26888	76261	0	76261
I2C-FIFO	26774	75943	0	75943
FSM	113	317	0	317

6. CONCLUSION

IP core designed in this paper is much efficient than any existing I^2C controller till present. To check and analyse the improvements done in this design, a thorough comparison has been done between generic open sourced RTL design for FPGA and our proposed modified ASIC version of master bus controller IP core. First, thorough simulation and elaboration has been done on Xilinx Tool to get the attributes of the generic module which includes large number of blocks and logic implementation. Then, 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.

6.1 Improvement Performance parameters

The parameters which are improved in ASIC design are as follows:

- Frequency of Operation
- Delay
- Power Consumption

Table7. Comparison Summary between generic open sourced I2C Master Controller RTL design and ASIC implemented modified I2C Master Controller IP Core

Parameter	Open Source I2C Design (opencores)	Proposed ASIC I2C design	Increase in Efficiency	COMMENTS
POWER - Static Power - Dynamic Power	160.03 mW 16 mW	1.589 uW 2.75 mW	10 times 5 times	-FPGA have more static power loss compared to ASICs design
WORST DELAY	Nearly 6nS	2.5 nS	58%	-The Worst path delay reduced by 3.5 Nano seconds.
MAX FREQUENCY	Max 185 MHz	400 MHz	Almost Thrice	-A tradeoff between max frequency and area overhead.
AREA REQUIMENT	Variable & depends upon FPGA unit	0.07 mm^2	Flexibility in sizing	- ASICs are the smaller & light designs.

6.2 IP Core

A final Gate level netlist has been generated along with .sdf delay file of the final Ip core design. Figure 10 shows overview of the netlist file of the firm IP of the designed I2C_Master_Controller Core.

```
module i2c_fifo(clk, rst, wr_en, rd_en, din, full, empty, dout);
  input clk, rst, wr_en, rd_en;
  input [14:0] din;
  output full, empty;
  output [14:0] dout;
  wire clk, rst, wr_en, rd_en;
  wire [14:0] din;
  wire full, empty;
  wire [14:0] dout;
  wire [9:0] rptr;
  wire [9:0] wptr;
  wire [14:0] \fifo_mem[0] ;
  wire [14:0] \fifo_mem[100] ;
  wire [14:0] \fifo_mem[101] ;
  wire [14:0] \fifo_mem[102] ;
  wire [14:0] \fifo_mem[103] ;
  wire [14:0] \fifo_mem[104] ;
  wire [14:0] \fifo_mem[105] ;
  wire [14:0] \fifo_mem[106] ;
  wire [14:0] \fifo_mem[107] ;
  wire [14:0] \fifo_mem[108] ;
  wire [14:0] \fifo_mem[109] ;
  wire [14:0] \fifo_mem[10] ;
  wire [14:0] \fifo_mem[110] ;
  wire [14:0] \fifo_mem[111] ;
  wire [14:0] \fifo_mem[112] ;
  wire [14:0] \fifo_mem[113] ;
  wire [14:0] \fifo_mem[114] ;
  wire [14:0] \fifo_mem[115] ;
  wire [14:0] \fifo_mem[116] ;
```

Verilog file   length : 2614172   lines : 51552

Figure 10. Overview of the final gate level netlist of firm IP core

6.3. Final Specification Summary of proposed I^2C Master Controller IP Core

Area: 0.076261 mm2 OR 76,261 um2

Power: 2.72 mW

Clock Frequency (max): 400 M Hz

Max Delay: 2.5 Nano Seconds

ACKNOWLEDGEMENTS

It is amongst the greatest clichés when writing a “big thanks”. We really do have to express our collective gratitude towards Dept. Of E&C, B.I.T, Bangalore for providing all the necessary research facilities required in this project. Special acknowledgements to the Head of the Department, Dr. K V Prasad (Professor, Dept. of E&C, B.I.T, Bangalore) for his kind cooperation thought the project.

REFERENCES

[1] Arvind Sahu, Dr. Ravi Shankar Mishra, Puran Gour, (2011)“An Implementation of I2C using VHDL for DATA surveillance”, International Journal on Computer Science and Engineering (IJCSE) ,Vol. 3 No. 5

[2] M.Jyothi, L.Ravi Chandra, M.Sahithi, S.Daya Sagar Chowdary, K.Rajasekhar, K.Purnima, (2012)“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

[3] Bollam Eswari, N.Ponmagal, K.Preethi, S.G.Sreejeesh, (2013) "Implementation of I2C Master Bus Controller on FPGA", International conference on Communication and Signal Processing, pp,1113-1116

[4] A.K. Oudjida, M.L. Berrandjia, R. Tiar, A. Liacha, K. Tahraoui, (2009) "FPGA Implementation of I2C & SPI Protocols: A Comparative Study" –IEEE Conference on Electronics, Circuits and Systems, pp. 507 – 510.

[5] Fang Zhao ; Daquan Deng ; Zhensheng Wang ; Huanzhao Liu , (2011) "Design of Schematic Mapping System Based on I2C and USB Bus" –First International Conference on Instrumentation, Measurement, Computer, Communication and Control, pp. 180-183.

[6] Oberg, J.; Wei Hu ; Irturk, A. ; Tiwari, M.,(2011) "Information flow isolation in I2C and USB" – International Conference on Design Automatic Conference(DAC), pp.254-259, 5-9

[7] Narendra Gupta, Vipin Gupta, (2013) "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.

[8] MR. J. J Patel, PROF B. H. Soni, (2013) " Design And Implementation Of I2C Bus Controller Using Verilog" Journal Of Information, Knowledge And Research In Electronics And Communication Engineering , Volume – 02, Issue - 02

[9] Marcello Dalpasso, Alessandro Bogliolo, and Luca Benini,(2000) "Hardware/Software IP protection." In Proceedings of the 37th Design Automation Conference,pp.593-596, June.

[10] Naagesh S. Bhat, (2011) "Design And ASIC Implementation Of DUC/DDC For Communication Systems" International Journal of VLSI design & Communication Systems (VLSICS) Vol.2, No.4

[11] Amandeep Singh, Balwinder Singh,(2012) "Microcontroller Based Testing of Digital IP-Core" International Journal of VLSI design & Communication Systems (VLSICS) Vol.3, No.2

[12] R.Uma And P. Dhavachelvan, (2012) "Synthesis Optimization For Finite State Machine Design In Fpgas" International Journal of VLSI design & Communication Systems (VLSICS) Vol.3, No.6.

AUTHORS

Vijaya Prakash A.M is working as Professor, Dept. of E&C, BIT, Bangalore. He completed his Ph.D. in VLSI in the Year 2012. He has many publications in Renowned conferences and International Journals. 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 and communication protocols.

Ankit K V is PG Scholar in Dept. of E&C, BIT, Bangalore, KA, India. He has received B.Tech (Hons) in ECE and now involved in active research in VLSI Mixed Mode Designs and Embedded Products etc.