International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in

Vol.2., Issue.6, 2014

RESEARCH ARTICLE

ISSN: 2321-7758

UART to SPI Core Implementation on different FPGA Devices

V.ASHOK[1], D.CHANDRA SEKHAR[2]

[1]Student of VLSI and ES Design, [2]Assistant Professor, ECE Department ,Prakasam Engineering College, Andhra Pradesh, Affiliated to JNTU-Kakinada

Article Received: 24/10/2014 | Article Revised on: 04/11/2014 | Article Accepted on:06/11/2014

V.ASHOK

D.CHANDRA SEKHAR

ABSTRACT

In electronic design a semiconductor intellectual property core, IP core, or IP block is a reusable unit of logic, cell, or chip layout design that is the intellectual property of one party. IP cores may be licensed to another party or can be owned and used by a single party alone.UART to SPI core is used in many Microcontrollers and SoC's. UART to SPI core can be used as a module in implementing a bigger system irrespective of ones choice of implementation platform.. The design is simulated on various FPGA Devices and power consumed is calculated. The results show the power consumed by the module in various devices.The results obtained can be used to select a type of FPGA device for implementation.The design can be used in SOC based devices, by directly integrating with other sub-blocks. Thereby, development time is highly reduced.

©KY Publications

.

INTRODUCTION

The importance for System-on-Chip (SOC) using an intellectual property (IP) is increasing in modern design methodology. The past design method is not suitable to design chip which operate on required function in given time. Therefore, using proper IP for requested specifications reduce design time and cope with time-to- market .So we designed a reusable UART IP for application of serial communication. A UART (Universal Asynchronous Receiver and Transmitter) is an integrated circuit which plays the most important role in serial communication. The UART contains a receiver (serial-to-parallel converter) and a transmitter (parallel-to-serial converter)[1]. It handles the conversion between serial and parallel data. Serial communication reduces the distortion of a signal, therefore makes data transfer between two systems separated in great distance possible. The advantages of UART systems are the simplicity of interconnection wiring and character transmission protocol and formats.

The System On Chip (SOC) based design involves ten's of modules integrated and formed as system. IP based design reduces the design time by directly integrating all the IP's developed by different vendors in the system. If the system needs to communicate with external devices like modem, a UART module may be need. At the same time, for internal on board communication with other devices SPI module may be needed. A UART device can be designed with different parameters. If the vendor changes the parameters, the entire design needs to be developed from scratch. For high performance systems, FPGA's sometimes use SPI to interface as a slave. In-system programmable AVR controllers (including blank ones) can be programmed using an SPI Interface

Architecture of interface

The UART to SPI module is implemented in Verilog HDL, which makes it platform independent and by changing the parameters in code, any design with specified parameters can be made available. Readily Available core, if SOC involves UART-SPI Interface. These SPI is the famous bus interface that is widely used in many SOC based design. It offers full duplex communication. A message handler is necessary for Communication between SPI and UART, which supports different data formats. The UART interface can be connected to serial communication devices like modem. The SPI Interface is connected to SPI Bus for serial communication with internal blocks/devices[2].

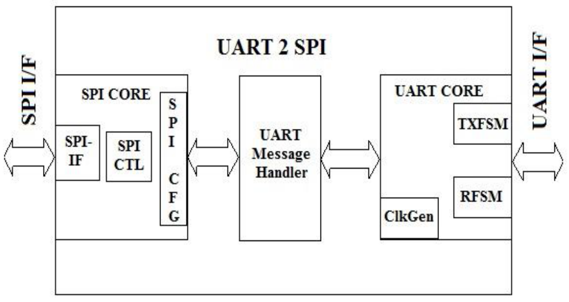


Fig1. block diagram of the UART to SPI core.

Architecture Consist of following Sub-blocks: A)UART CORE. B)SPI CORE .C)UART Message Handler[3]

A. UART Core

It translates data between parallel and serial forms. This core includes 3 parts.

i) RXFSM: This block monitor the UART receives port (uart_rxd) and decode it into 8 bit data format.

ii)TXFSM: This block translate 8 Bit Data into serial UART bit frame format and drive it into UART Transmit Port (uart_txd)[4].

iii) CLK-GEN: This block includes a clock divider circuit to generate required baud-clock required to sample/drive the UART interface signal.

16x Baud clock formula is = System-Clock (in Hz) / (2 + Configured Baud Register value).

B. UART_SPI CNTRL

This block includes a state machine and decodes

i) wm <addr> <data> <newline> command into register write towards SPI core with <Addr> <Data>

ii) rm <addr> <newline> command into a register read access towards SPI core and transmit back the read data received from SPI interface[5].

C. Serial Peripheral Interface(SPI)

Serial Peripheral Interface, often shortened as SPI (pronounced as *spy*, or*ess-pee-eye*), is a synchronous serial data transfer protocol named by Motorola. Here two or more serial devices are connected to each other in full-duplex mode. The devices connected to each other are either *Master* or *Slave*. In a SPI link there could as many Masters and Slaves as required, but it's very rare to find more than one Master in a SPI link[6].

The *Master* device is the one which initiates the connection and controls it. Once the connection is initiated, then the *Master* and one or more *Slave(s)*can transmit and/or receive data. As mentioned earlier, this is a full-duplex connection, which means that *Master* can send data to *Slave(s)* and the *Slave(s)* can also send the data to the *Master* at the same time.

Now that we have a basic knowledge of what SPI is, let's look into the operation of SPI Bus. The SPI operation is based upon shift registers. Every device, whether *Master* or *Slave* has an 8-bit shift register inside it. The size of the shift register could be more than 8-bit as well (like 10-bit, 12-bit, etc), but it should be the same for both *Master* and *Slave*, and the protocol should support it. The *Master* and *Slave* are connected in such a way that the two shift registers form an inter-device circular buffer. These shift registers operate in Serial-In/Serial-Out (SISO) fashion. The output of the *Master's* shift register is connected to the input of the *Slave's* shift register; and the output of the *Slave's* shift register is connected to the input of *Master's* shift register. This makes the connection operate like a circular/ring buffer. Don't bother about the names MISO, MOSI and SCK now. We will discuss about them a little later in this post.

As mentioned earlier, SPI is a synchronous serial data transfer protocol, which means that there must be a clock to synchronize the data transfer. It has also been stated that the *Master* is responsible for initiating and controlling the connection.

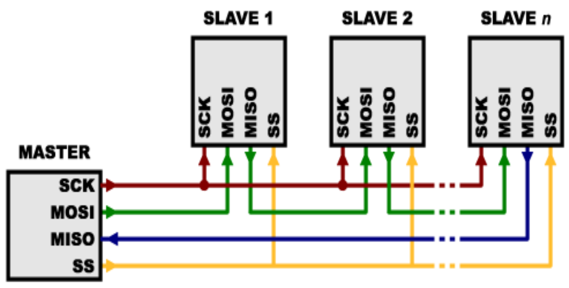


Fig 3.SPI Master-Slave Configuration.

VERIFICATION

The Device Under Test (DUT) is UART-SPI Core. It is verified by running test cases and the SPI salve device selected is ST Serial Flash Memory. In Directed verification, the Verification Environment has mechanism to send the Stimulus to DUT and collect the responses and check the responses. The Stimulus is generated in Tests case. Directed testbenchs may also use a limited amount of randomization, often by creating random data values rather than simply filling in each data element with a predetermined value. Each test case verifies specific feature of the design. This becomes tedious when the design complexity increases. As circuit complexity increases, it becomes more difficult to create patterns that fully exercise the design. Test case maintenance become harder and time consuming[7].

A **Bus Functional Model** or BFM (also known as a Transaction Verification Models or TVM)[8] is a non-synthesizable software model of an integrated circuit component having one or more external buses. The emphasis of the model is on simulating system bus transactions prior to building and testing the actual hardware. BFM's are usually defined as tasks in Hardware description languages (HDLs), which applies stimulus to the design under test/verification via complex waveforms and protocols. A BFM is typically written in an HDL language such as verilog, VHDL etc.On one side, it drives and samples low-level signals according to the bus protocol. On the other side, tasks are available to create and respond to bus transactions[9].

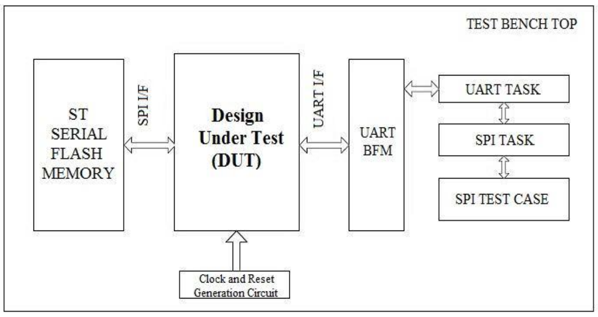


FIG 4.VERIFICATION FLOW

BFMs are often used as reusable building blocks to create simulation test benches, where the signal ports on a design under test are connected to the appropriate BFMs in the test bench for the purpose of simulation[10].

UART Bus Functional Model (BFM): This block manages the UART protocol format translation from 8bit data to serial format and vice-versa.

Flash Memory: It act a SPI Slave device, which is connected through SPI Bus.

Assert Property: The assert statement is used to enforce a property as a checker[11]. When the property for the assert statement is evaluated to be true, the pass statements of the action block are executed. Otherwise, the fail statements of the action block are executed. Assert Property has Tasks.2 types of task in this project are[12],

SPI Task: This module includes various tasks to configure the on-chip SPI module.

UART Task: This module includes various tasks to configure the on-chip UART module[13].

RESULTS

A. *SYNTHESIS BLOCK DIAGRAMS:*

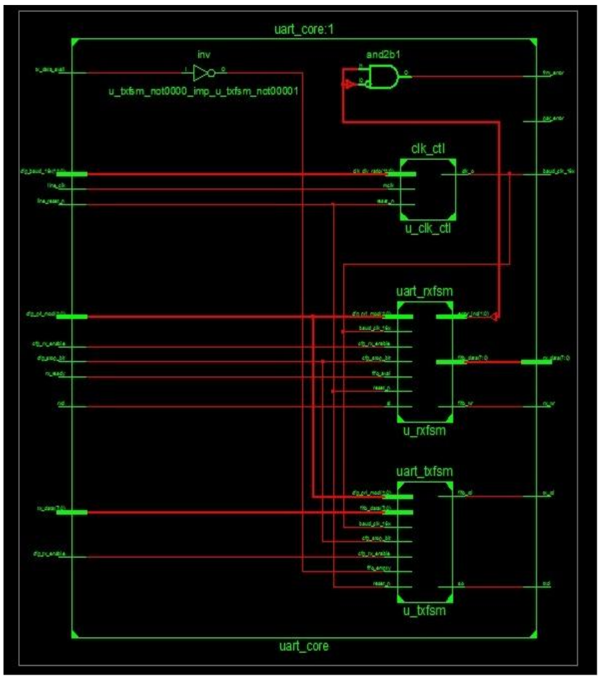


Fig 5.UART Core top module

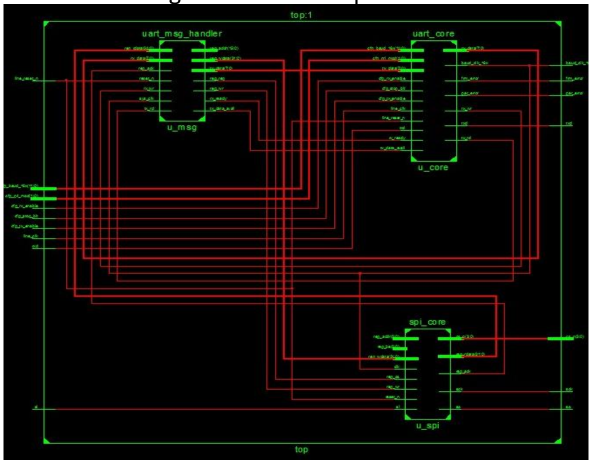


Fig 6.UART to SPI Core Top level module

B. *SIMULATION DIAGRAMS:*

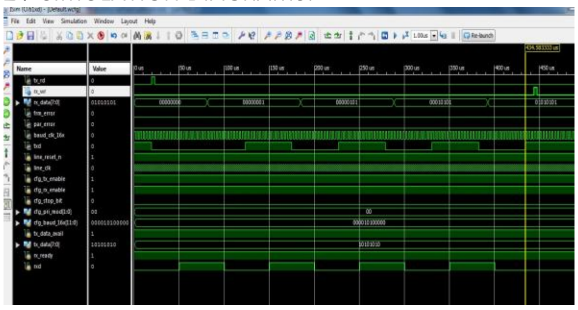


Fig 7.Uart Receiver (without parity)

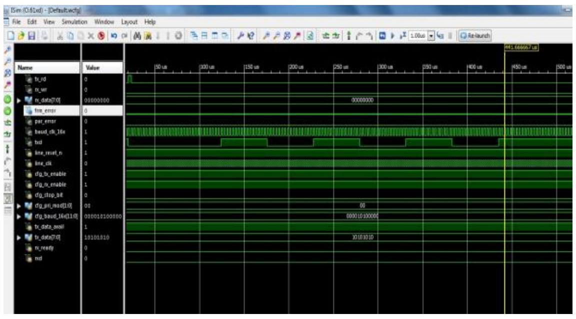


Fig 8.Uart Transmitter(without Parity)

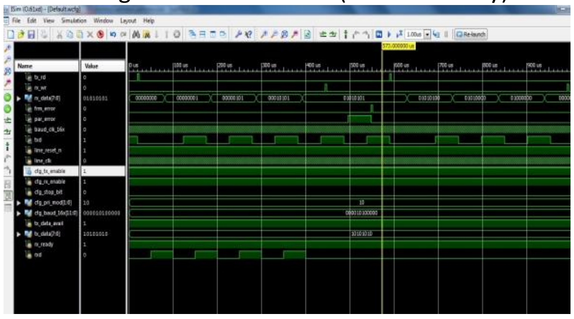


Fig 9.Uart with even Parity

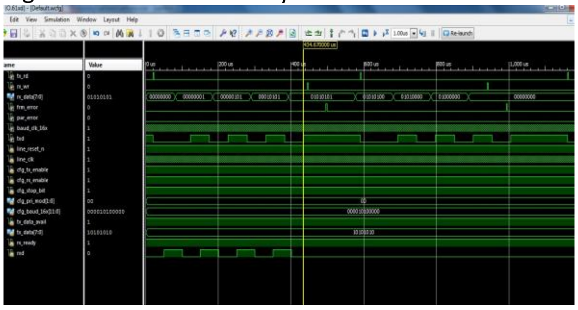


Fig 10.Uart with 2 stop bits

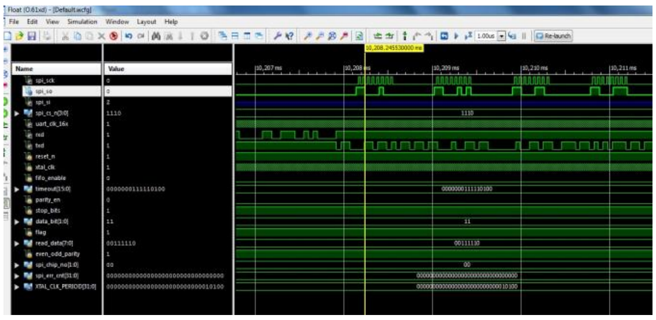


Fig 11.Uart to SPI Memory Write Operation

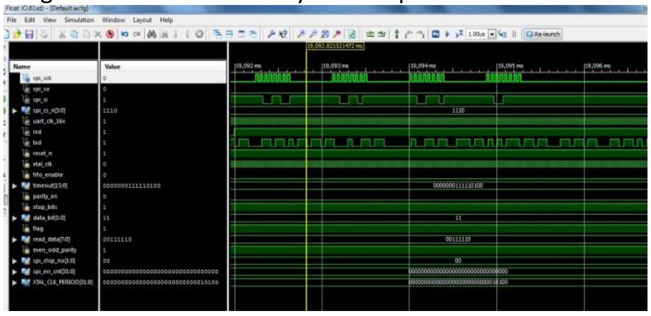
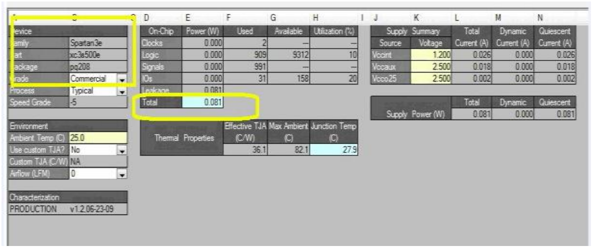


Fig 12.Uart to SPI Memory Read Operation

C.POWER CONSUMPTION:



evice	
amily	Spartan3e
art	xc3s500e
ackage	pq208
rade	Commercial
Process	Typical
Speed Grade	-5

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0

Characterization	
PRODUCTION	v1.2.06-23-09

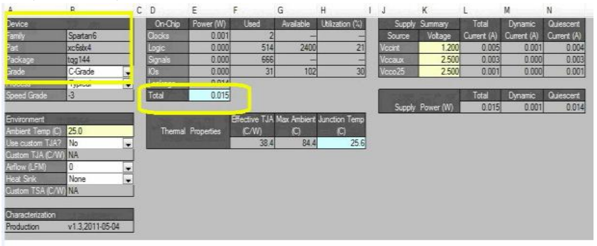
On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	2	---	---
Logic	0.000	909	9312	10
Signals	0.000	991	---	---
IOs	0.000	31	158	20
Leakage	0.081			
Total	0.081			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	36.1	82.1	27.9

Supply Summary Source	Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.200	0.026	0.000	0.026
Vccaux	2.500	0.018	0.000	0.018
Vcco25	2.500	0.002	0.000	0.002

	Total	Dynamic	Quiescent
Supply Power (W)	0.081	0.000	0.081

Fig 13. Power Consumption Report For Spartan 3e(PQ 208)



Device	
Family	Spartan6
Part	xc6slx4
Package	tqg144
Grade	C-Grade
Process	Typical
Speed Grade	-3

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0
Heat Sink	None
Custom TSA (C/W)	NA

Characterization	
Production	v1.3,2011-05-04

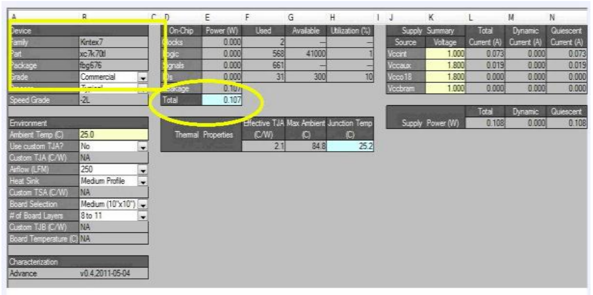
On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.001	2	---	---
Logic	0.000	514	2400	21
Signals	0.000	666	---	---
IOs	0.000	31	102	30
Leakage	0.014			
Total	0.015			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	38.4	84.4	25.6

Supply Summary Source	Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.200	0.005	0.001	0.004
Vccaux	2.500	0.003	0.000	0.003
Vcco25	2.500	0.001	0.000	0.001

	Total	Dynamic	Quiescent
Supply Power (W)	0.015	0.001	0.014

Fig 14. Power Consumption Report For Spartan 6(tqg 144)



Device	
Family	Kintex7
Part	xc7k70tl
Package	fbg676
Grade	Commercial
Process	Typical
Speed Grade	-2L

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	8 to 11
Custom TJB (C/W)	NA
Board Temperature (C)	NA

Characterization	
Advance	v0.4,2011-05-04

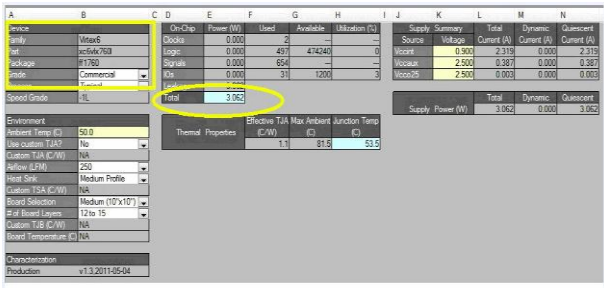
On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	2	---	---
Logic	0.000	568	41000	1
Signals	0.000	661	---	---
IOs	0.000	31	300	10
Leakage	0.107			
Total	0.107			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	2.1	84.8	25.2

Supply Summary Source	Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.000	0.073	0.000	0.073
Vccaux	1.800	0.019	0.000	0.019
Vcco18	1.800	0.000	0.000	0.000
Vccbram	1.000	0.000	0.000	0.000

	Total	Dynamic	Quiescent
Supply Power (W)	0.108	0.000	0.108

Fig 15. Power Consumption Report For Kintex (FBG 676)



Device	
Family	Virtex6
Part	xc6vlx760l
Package	ff1760
Grade	Commercial
Process	Typical
Speed Grade	-1L

Environment	
Ambient Temp (C)	50.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	12 to 15
Custom TJB (C/W)	NA
Board Temperature (C)	NA

Characterization	
Production	v1.3,2011-05-04

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	2	---	---
Logic	0.000	497	474240	0
Signals	0.000	654	---	---
IOs	0.000	31	1200	3
Leakage	3.062			
Total	3.062			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	1.1	81.5	53.5

Supply Summary Source	Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	0.900	2.319	0.000	2.319
Vccaux	2.500	0.387	0.000	0.387
Vcco25	2.500	0.003	0.000	0.003

	Total	Dynamic	Quiescent
Supply Power (W)	3.062	0.000	3.062

Fig 16. Power Consumption Report For Virtex(ff 1760)

CONCLUSION

We have designed our UART to SPI Interface available as IP which can be used in building larger systems[14] like SoC.The IP is simulated on different FPGA Devices and comparision is made for power consumption[15] and size occupied.

FPGA device family	**Power Consumption(mW)**
Spartan 3e(package PQ 208)	81
Spartan 6(package TQG 144)	15
kintex 7(package FBG 676)	107
Virtex 6(package FF 1760)	3062

REFERENCES

[1]. B.Roy,Platform-Independent Customizable UART Soft-Core ,Intelligent Systems, Modelling and Simulation (ISMS), IEEE Third International Conference 2012.p 692-694.

[2]. Dinesh Annayya, UART to SPI Specification available :http://www.opencores.org.

[3]. MicroSemi notes:AC 327 UART to SPI Interface Design example available: *www.actel.com/documents/**UART_to_SPI**_AN.**p df***

[4]. http://opencores.org/usercontent,doc,1359617 335

[5]. http://opencores.org/project,uart2spi

[6]. http://en.wikipedia.org/wiki/Bus_Functional_M odel

[7]. http://www.cyclopaedia.info/wiki/Transaction-Verification-Model

[8]. http://www.cyclopaedia.es/wiki/Transaction_V erification_Model

[9]. www. ijoer.in/Vol1.Issue.2/Ikram%20271-274.pdf

[10]. Verilog HDL by Samir Palnitkar Publisher: Prentice Hall PTR (January 15, 1996)

[11]. Parag k..Lala Introduction to Logic Circuit Testing Morgan and Claypool Publishers 2009.

[12]. Xilinx Synthesis Overview available at *www.**xilinx**.com/support/**xilinx**11/**ise**_c_using_x st_for_**synthesis**.htm*

[13]. ISIM in-dept available :*www.**xilinx**.com/itp/**xilinx**10/isehelp/ism_p_ru nning_**simulation**_ise.htm.*

[14]. Blessington, T.P; Murthy, B.B. ; Ganesh, G.V. ; Prasad, T.S.R. Devices, Circuits and Systems (ICDCS), 2012 International Conference.p:673-677

[15]. http://www.eeherald.com/section/design-guide/esmod12.html