IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 3, June-July, 2013
ISSN: 2320 - 8791
www.ijreat.org

Design of NoC Router for SoC Architecture

Ulaganathan.J[1], Ahmed Basha[2], H.Devanna[3], K.Sudhakar[4]
[1,2,3,4] *Department of Electronics and Communication Engineering, SJCET-Kurnool , A.P , India*

Abstract **-- The focus of this Paper is the actual implementation of Network Router and verifies the functionality of the five port router for network on chip using the latest verification methodologies, Hardware Verification Languages(like Open Verification Methodologies) and EDA tools and qualifies the Design for Synthesis and implementation. This Router design contains Four output ports and one input port, it is packet based Protocol. This Design consists of Registers, FSM and FIFO's. The Verification goes on it finds functional coverage of the Network Router by using System Verilog-OVM.**

Keywords **-- System Verilog, Fictional Coverage, Assertions, Randomization, FIFO, FSM, Network-On-Chip(NoC), System-On-Chip(SoC), Verification Methodologies.**

I. INTRODUCTION

90% of ASIC re-spins are due to Functional bugs. As the functional verification decides the quality of the silicon, we spend 60% of the design cycle time only for the verification/simulation. In order to avoid the delay and meet the TTM, we use the latest verification methodologies and technologies and accelerate the verification process. This project helps one to understand the complete functional verification process of complex ASICs and SOC's and it gives opportunity to try the latest verification methodologies, programming concepts like Object Oriented Programming of Hardware Verification Languages and sophisticated EDA tools, for high quality verification.

The challenge of verifying a large design is growing exponentially. There is a need to define new methods that makes functional verification easy. Several strategies in the recent years have been proposed to achieve good functional verification with less effort. Recent advancement towards this goal is methodologies. The methodology defines a skeleton over which one can add flesh and skin to their requirements to achieve functional verification. OVM (open verification methodology) is one such efficient methodology and best thing about it is, free and reuse. This OVM is built on System Verilog and used effectively to achieve maintainability, reusability, speed of verification etc. This Paper is aimed at building a reusable testbench for verifying Router Protocol by using system verilog and OVM.

For most home users, they may want to set-up a LAN (Local Area Network) or WLAN (wireless LAN) and connect all computers to the Internet without having to pay a full broadband subscription service to their ISP for each computer on the network. In many instances, an ISP will allow you to use a router and connect multiple computers to a single Internet connection and pay a nominal fee for each additional computer sharing the connection. This is when home users will want to look at smaller routers, often called broadband routers that enable two or more computers to share an Internet connection. Within a business or organization, you may need to connect multiple computers to the Internet, but also want to connect multiple private networks not all routers are created equal since their job will differ slightly from network to network. Additionally, you may look at a piece of hardware and not even realize it is a router.

II. ROUTER DESIGN PRINCIPLES

Given the strict contest deadline and the short implementation window we adopted a set of design principles to spend the available time as efficiently as possible. This document provides specifications for the Router is a packet based protocol. Router drives the incoming packet which comes from the input port to output ports based on the address contained in the packet. The router is a" Network Router" has a one input port from which the packet enters. It has four output ports where the packet is driven out. Packet contains 3 parts. They are Header, data and frame check sequence. Packet width is 8 bits and the length of the packet can be between 1 byte to 63 bytes. Packet header contains three fields DA, Data and Length. Destination address (DA) of the packet is of 8 bits. The switch drives the packet to respective ports based on this destination address of the packets. Each output port has 8- bit unique port address. If the destination address of the packet matches the port address, then switch drives the packet to the output port, Length of the data is of 8 bits and from 0 to 63. Length is measured in terms of bytes. Data should be in terms of bytes and can take anything. Frame check sequence contains the security check of the packet. It is calculated over the header and data. The communication on network on chip is carried out by means of router, so for implementing better NOC, the router should be efficiently designed. This router supports four parallel connections at the same time.

It uses store and forward type of flow control and FSM Controller deterministic routing which improves the performance of router. The switching mechanism used here is packet switching which is generally used on network on chip. In packet switching the data transfers in the form of packets between co-operating routers and Independent routing decision is taken. The store and forward flow mechanism is best because it does not reserve channels and thus does not lead to idle physical channels. The arbiter is of rotating priority scheme so that every channel once get chance to transfer its data. In this router both input and output buffering is used so that congestion can be avoided at both sides.

Features

- Full duplex synchronous serial data transfer.
- Variable length of transfer word up to 64 bytes.
- HEADER is the first data transfer.
- Fully static synchronous design with one clock domain
- Technology independent VERILOG Fully synthesizable.
- ROUTER is a Synchronous protocol.

The clock signal is provided by the master to provide synchronization. The clock signal controls when data can change and when it is valid for reading. Since ROUTER is synchronous, it has a clock pulse along with the data. RS-232 and other asynchronous protocols do not use a clock pulse, but the data must be timed very accurately. Since ROUTER has a clock signal, the clock can vary without disrupting the data. The data rate will simply change along with the changes in the clock rate. As compared with its counterpart I2C, ROUTER is more suited for data stream applications. Communication between IP's.

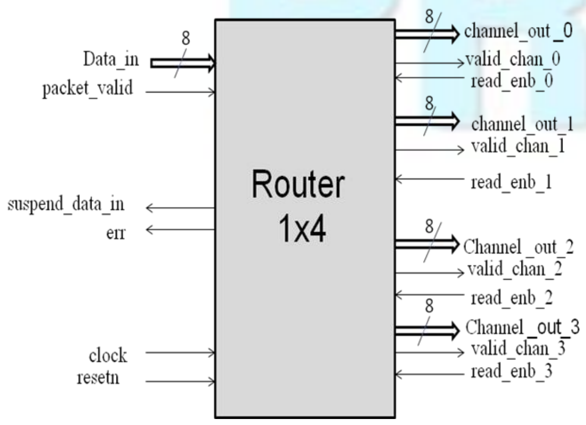


Figure1: Block Diagram of Router Protocol

III. Operation

The Five Port Router Design is done by using of the three blocks. The blocks are 8-Bit Register, Router Controller and output block. The router controller is design by using FSM design and the output block consists of four FIFO's combined together. The FIFO's store data packets and when you want to send data that time the data will read from the FIFO's. In this router design has four outputs i.e. 8-Bit size and one 8-bit data port. It is used to drive the data into router. we are using the global clock, reset signals, error signal and suspended data signals are the output's of the router. The FSM controller gives the error and SUSPENDED_DATA_IN signals.

These functions are discussed clearly in below FSM description. The ROUTER can operate with a single master device and with one or more slave devices. If a single slave device is used, the RE (read enable) pin may be fixed to logic low if the slave permits it. Some slaves require the falling edge (HIGH→LOW transition) of the slave select to initiate an action such as the mobile operators, which starts conversion on said transition. With multiple slave devices, an independent RE signal is required from the master for each slave device.

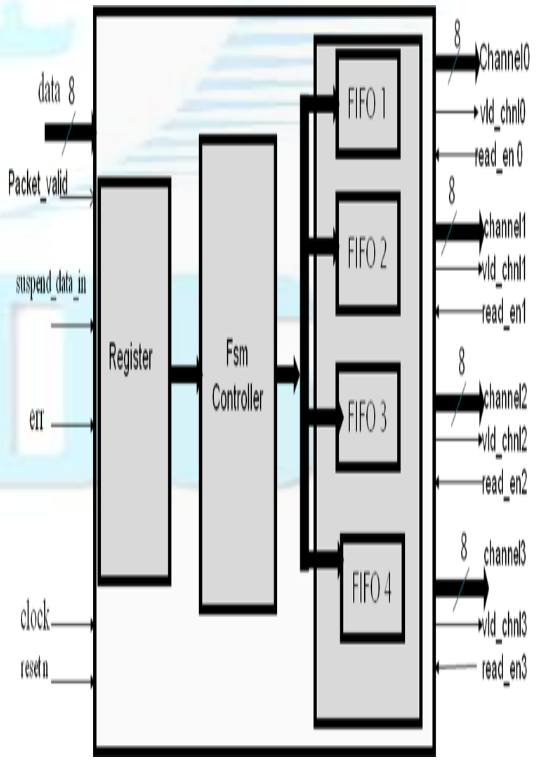


Figure2: Internal Structure of Protocol

IV. Verification Methodologies

Verification:

Verification is not a testbench, nor is it a series of testbenches. Verification is a process used to demonstrate that the intent of a design is preserved in its implementation. In this chapter, I introduce the basic concepts of verification, from its importance and cost, to making sure you are verifying that you are implementing what you want. The differences are presented between various verification approaches as well as the difference between testing and verification. It is also showed that how verification is key to design reuse, and challenges of verification reuse.

Formal verification, property checking, functional verification, and rule checkers verify different things because they have different origin and reconvergence points. In its most common use, equivalence checking compares two netlists to ensure that some netlist post-processing, such as scan chain insertion, clock-tree synthesis or manual modification1, did not change the functionality of the circuit. Property checking is a more recent application of formal verification technology. In it, assertions or characteristics of a design are formally proven or disproved. The main purpose of functional verification is to ensure that a design implements intended functionality.

Importance of verification

Today, in the era of multi-million gate ASICs and FPGAs, reusable intellectual property (IP), and system-on-a-chip (SoC) designs, verification consumes about 70% of the design effort. Design teams, properly staffed to address the verification challenge, include engineers dedicated to verification. The number of verification engineers can be up to twice the number of RTL designers.

Functional coverage

Functional coverage is another technology to help ensure that a bad design is not hiding behind passing testbenches. Although this technology has been in use at some companies for quite some time, it is a recent addition to the arsenal of general-purpose verification tools. Functional coverage records relevant metrics (e.g., packet length, instruction opcode, buffer occupancy level) to ensure that the verification process has exercised the design through all of the interesting values. Whereas code coverage measures how much of the implementation has been exercised, functional coverage measures how much of the original design specification has been exercised.

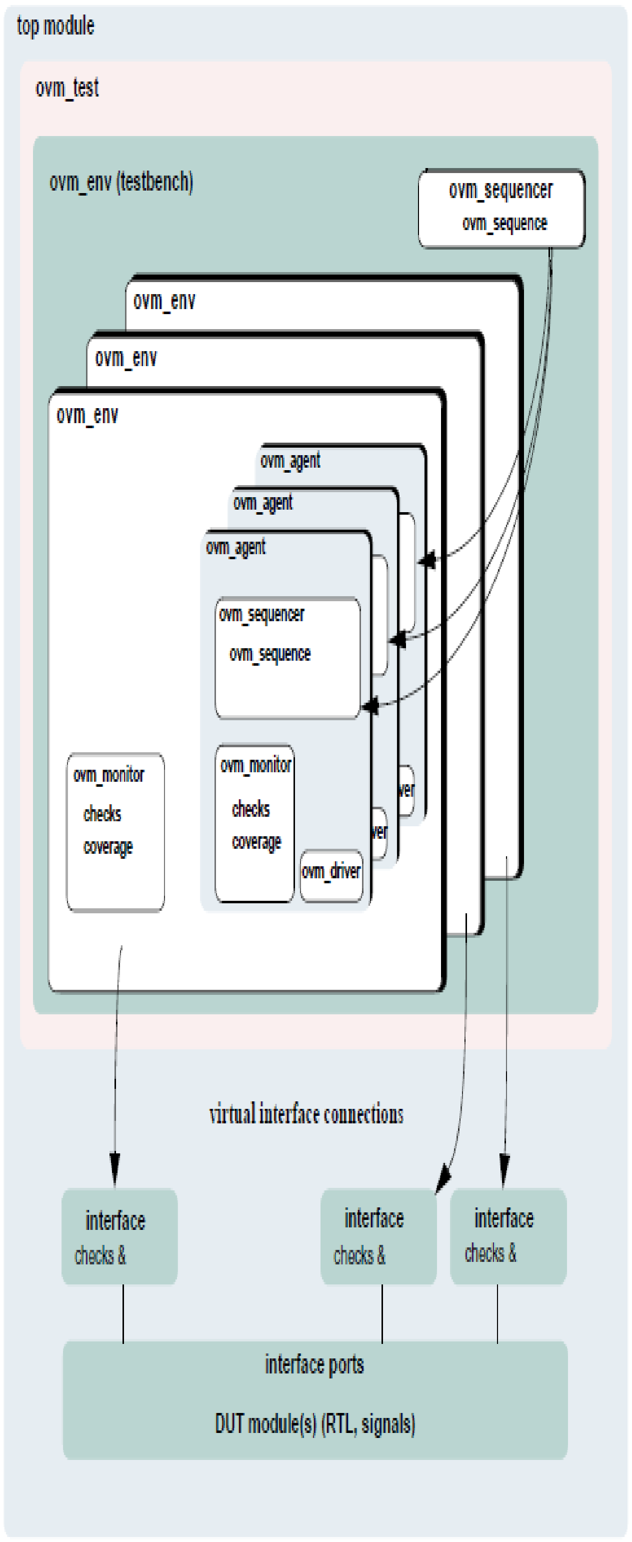


Figure 3: Complete block diagram of Verification Environment

V. TEST BENCH ARCHITECTURE

Test Plan:

A collection of all test specifications for a given area. The Test Plan contains a high-level overview of what is tested and what is tested by others for the given feature area. The test plan is implemented using OVM test component. Generally the method to write a test case is to define a base test in which all the environment is instantiated and all the common configurations used for all the test cases are written.

Test Cases:

Control test case

Control test case is mainly targeted for the usb section of the bridge. Its purpose is to check whether the device is responding to the standard request or not and to check the control endpoints of the device. It generates random standard requests to the Router devices and look for the response for the device. It also checks the operation of the endpoint 0 controller in the device.

Data Receive Test Case

This test case is used to verify the functionality of the master receiver of the Routerdevice. This test case first configures the device for use by using standard requests, issues the command to receive the data from the external i2c device and finally retrives the data from the master reciever FIFO by using data read commands. It uses standard sequences, write register sequence, data read sequence from the sequence library.

Slave Data Transmit Test Case

This test case is to verify the slave transmitter of the Router section, but any way this test case should make sure some data in slave transmit FIFO. So the test case will first calls standard commands to set the device up , then it has to write some data on the slave transmit FIFO and finally slave is ready to transmit the data.The transmission happens when external master initiates the transfer. It uses standar sequences, data write sequences, register write sequences from the sequence library.

Slave Data Receive Test Case

This test case is used to verify the functionality of the slave receiver of the Router device. This test case first configures the device for use by using standard requests, slave receives the data from the external i2c device and stores it in the slave receiver FIFO .Then agent reads the FIFO by using data reads commands. It uses standard sequences, write register sequence, data read sequence from the sequence library.

VI. RESULTS & COVERAGE REPORTS

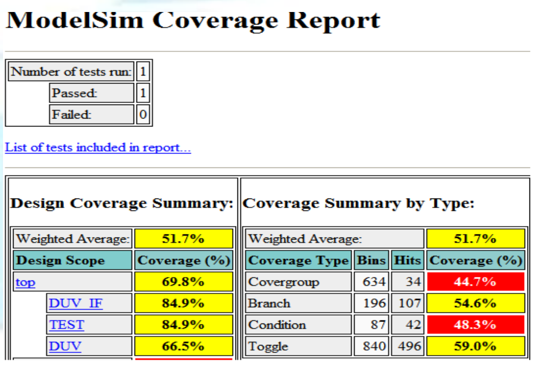
Code Coverage

The tests are written according the test plan and the tests are made into a single regression which contains all the tests and a Perl script is written to run all the tests. This script can also run with the coverage option to see the coverage results. It creates a html page which contains the results of the tests that are run in that regression. When this script is run with the coverage option, it creates a directory with name “report” and dumps all the coverage html reports generated by the Questa tool. The extent of verification cannot be judged manually.

The Questa tool offers coverage aspects on the written RTL code. Code coverage measures the amount of HDL code that has been exercised by all the tests. It not only checks the coverage in terms of lines covered, but the states covered in state machines, the values of the signals that are toggled etc. By running the tests in the testplan the coverage that is attained is as follows

A total of 81.8% overall coverage was attained which says that 81.8% of the DUT was verified. The rest of the coverage percentage is due to some part of the code which is kept for some invalid signalling. This percentage of the missed coverage is due to the conditions that has different possibilities like a condition ((x==1) or (y==1)). Tool creates the possible combinations of the condition as 00,01,10,11. Attaining all the conditions is cumbersome to achieve. Another reason is that the invalid FSM transitions that are picked up by the tool. The tool extracts the state machine from the HDL code. In that process some invalid transitions are created that are not present in the state machine which is designed. We can also eliminate the transitions from the coverage and get the coverage 100%.

Figure 4: Functional Coverage of Router only Directed Test cases



ModelSim Coverage Report

Number of tests run:	1
Passed:	1
Failed:	0

List of tests included in report...

Design Coverage Summary:

Weighted Average:	51.7%
Design Scope	**Coverage (%)**
top	69.8%
DUV_IF	84.9%
TEST	84.9%
DUV	66.5%

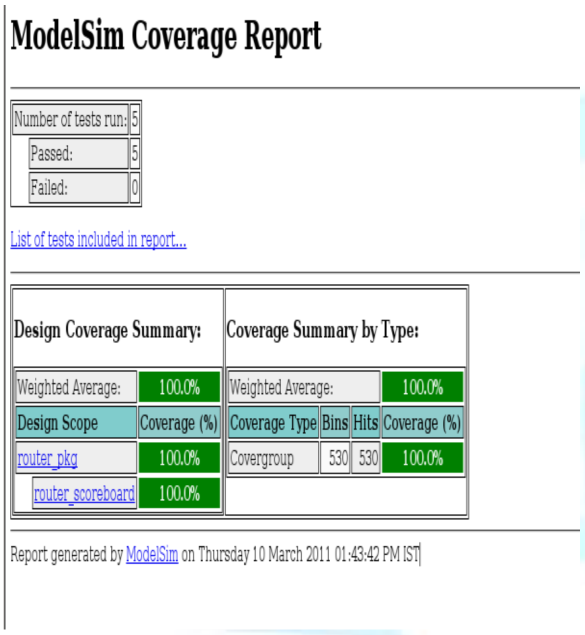
Coverage Summary by Type:

Weighted Average:			51.7%
Coverage Type	**Bins**	**Hits**	**Coverage (%)**
Covergroup	634	34	44.7%
Branch	196	107	54.6%
Condition	87	42	48.3%
Toggle	840	496	59.0%

Functional Coverage

Functional coverage is the determination of how much functionality of the design has been exercised by the verification environment. Code Coverage will give information about how many lines are executed, how many times expressions, branches executed. This coverage is collected by the simulation tools. Both of them have equal importance in the verification. 100% functional coverage does not mean that the DUT is completely exercised and vice-versa. Verification engineers will consider both coverages to measure the verification progress.

In figure 4 only one test case has been passed so the coverage is 51.7% and out of 634 bins only 34 bins has been hitted.Inorder to increase the coverage , increase the testcases for which the bins hasnot been covered this was shown in figure 5 .Atlast in figure 5 testcases has been passed and all the bins(530) has been covered so,finally 100% coverage has been reached.



ModelSim Coverage Report

Number of tests run:	5
Passed:	5
Failed:	0

List of tests included in report...

Design Coverage Summary:

Weighted Average:	100.0%
Design Scope	Coverage (%)
router_pkg	100.0%
router_scoreboard	100.0%

Coverage Summary by Type:

Weighted Average:			100.0%
Coverage Type	Bins	Hits	Coverage (%)
Covergroup	530	530	100.0%

Report generated by ModelSim on Thursday 10 March 2011 01:43:42 PM IST

Figure 5: Functional Coverage of Router With Constrained Random Verification

Simulation Results

The below figures shows the simulation results of test cases applied to the DUT. Figure 6.shows the response of the device for the control test case at the USB interface. Figure 6. Shows the master transmitter sending random data to the external slave device.

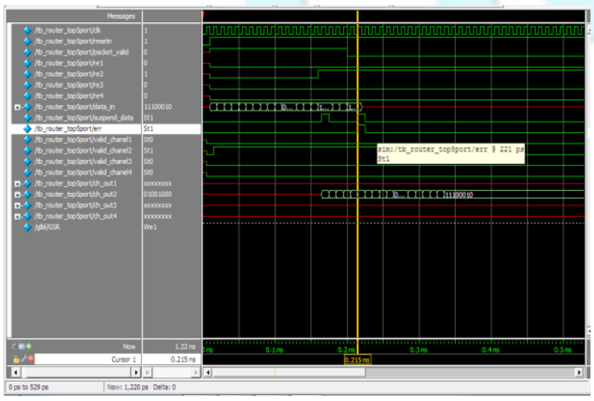


Figure 6: FSM Simulation Results

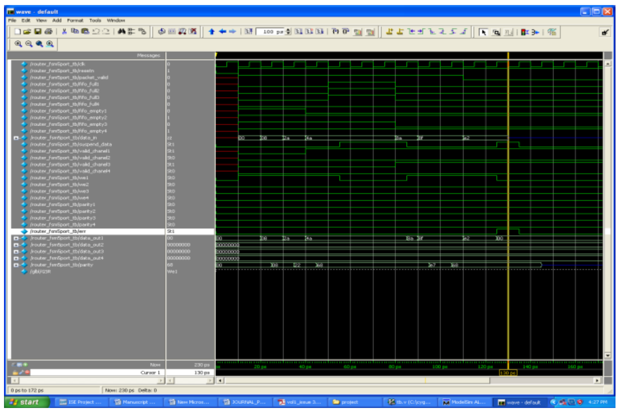


Figure 7: Simulation Results of Router Protocol

Data_in[1:7] = input =deferent data packets

Data_in[0] = 11111001

Output :

Ch_out1 = 8'bXXXXXXXX

Ch_out2 = data_in,

Ch_out3 = 8'bXXXXXXXX.

After **80ns** again applying inputs immediately output obtained, because of no delay elements.

Data_in[1:7] = input =deferent data packets

Data_in[0] = 11111000

Output :

Ch_out2 = 8'bXXXXXXXX,

Ch_out1 = data_in,

Ch_out3 = 8'bXXXXXXXX

Synthesis Report:

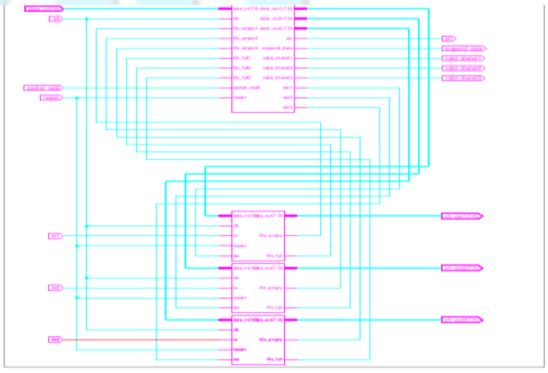


Figure 8 : Synthesis results of Router for NoC.

VII. APPLICATIONS

When multiple routers are used in interconnected networks, the routers exchange information about destination addresses, using a dynamic routing protocol. Each router builds up a table listing the preferred routes between any two systems on the interconnected networks. A router has interfaces for different physical types of network connections, (such as copper cables, fiber optic, or wireless transmission). It also contains firmware for different networking protocol standards. Each network interface uses this specialized computer software to enable data packets to be forwarded from one protocol transmission system to another. Routers may also be used to connect two or more logical groups of computer devices known as subnets, each with a different sub - network address. The subnet addresses recorded in the router do not necessarily to map directly to the physical interface connections.

VIII. CONCLUSION

As the functional verification decides the quality of the silicon, we spend 60% of the design cycle time only for the verification/simulation. In order to avoid the delay and meet the TTM, we use the latest verification methodologies and technologies and accelerate the verification process. This project helps one to understand the complete functional verification process of complex ASICs an SoC's and it gives opportunity to try the latest verification methodologies, programming concepts like Object Oriented Programming of Hardware Verification Languages and sophisticated EDA tools, for the high quality verification.

In this project I have verified the functionality of the ROUTER with the latest Verification methodology i.e. System VERILOG and observed the code coverage and functional coverage of ROUTER by using cover points and different test cases (like constrained, weighted and directed test cases). By using these test cases I had improved the functional coverage of the ROUTER. In previous we using Verilog for verify the functionality here we can't find that results are correct or not by using verification methodologies we get 100% functional coverage. In this project I used one master and four slaves to monitor the ROUTER. Thus the functional coverage of the ROUTER was improved.

The results shows that System Verilog methodology can be used to make reusable test benches successfully. Large part of the test bench is made reusable over multiple projects. even though this reusablity is limited to the interfaces. A large class of devices that are build on these inerfaces can be verified successfully. Once these components are made the amount of time required to build test benches for other projects can be reduced a lot.

REFERENCES

1. D.Chiou,"MEMOCODE2011Hardware/SoftwareCoDesignContest",https://ramp.ece.utexas.edu/redmine/Attachments/ DesignContest.pdf
2. Xilinx,"ML605HardwareUserGuide",http://www.xilinx.com/support/documentation/boardsand its/ug534.pdf
3. Xilinx,"LogiCOREIPProcessor Local Bus (PLB) v4.6", http://www.xilinx.com/support/documentation/ip documentation/plb v46.pdf
4. "Application Note: Using the Router Interface to Communicate Motorola, ANN91/D Rev. 1, 01/2001. Cisco Router OSPF: Design& Implementation Guide, Publisher: McGraw-Hill
5. "Nortel Secure Router 4134", Nortel Networks Pvt. Ltd.
6. "LRM", IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language – IEEE STD 1364-1995.
7. Thomas, Donald, Moorby, Phillip "The Verilog Hardware Description Language" Kluwer Academic Publishers, Norwell.
8. Janick Bergerdon, "Writing Testbenches: Functional Verification of HDL Models".
9. Verilog HDL "A Guide to Digital design and Synthesis" by Samir Palnitkar.
10. Programmable Logic Design Quick Start Hand book by Karen Parnell and Nick Mehta.
11. The Complete Verilog Book by Vivek Sagdeo Sun Micro Systems, Inc.
12. Rajeev Madhavan, Verilog HDL reference Guide, Automata Publishing company, CA ISBN 0-9627488-4-6
13. Stuart Sutherland, Verilog HDL 2.0 Language Reference Guide.