iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** X **Month of publication:** October 2017

DOI: http://doi.org/10.22214/ijraset.2017.10192

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue X, October 2017- Available at www.ijraset.com

Design and Verification of Router 1x3 Using UVM

Salma Jabeen[1], Asarafuddin M D[2]

Abstract: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 has a one input port from which the packet enters. It has three output ports where the packet is driven out.In this project we are atomizing the functions of the Router by writing the code in VERILOG and simulating it in QUESTASIM. The functionality of the design is verified by using the latest Universal verification methodologies (UVM) with different test cases. The Verification goes on with which it finds functional coverage, code coverage and functional verification of the router 1x3 RTL design for the better optimum design.

Keywords: router 1x3, QUESTASIM, XILINX ISE, Verilog, UVM, Coverage

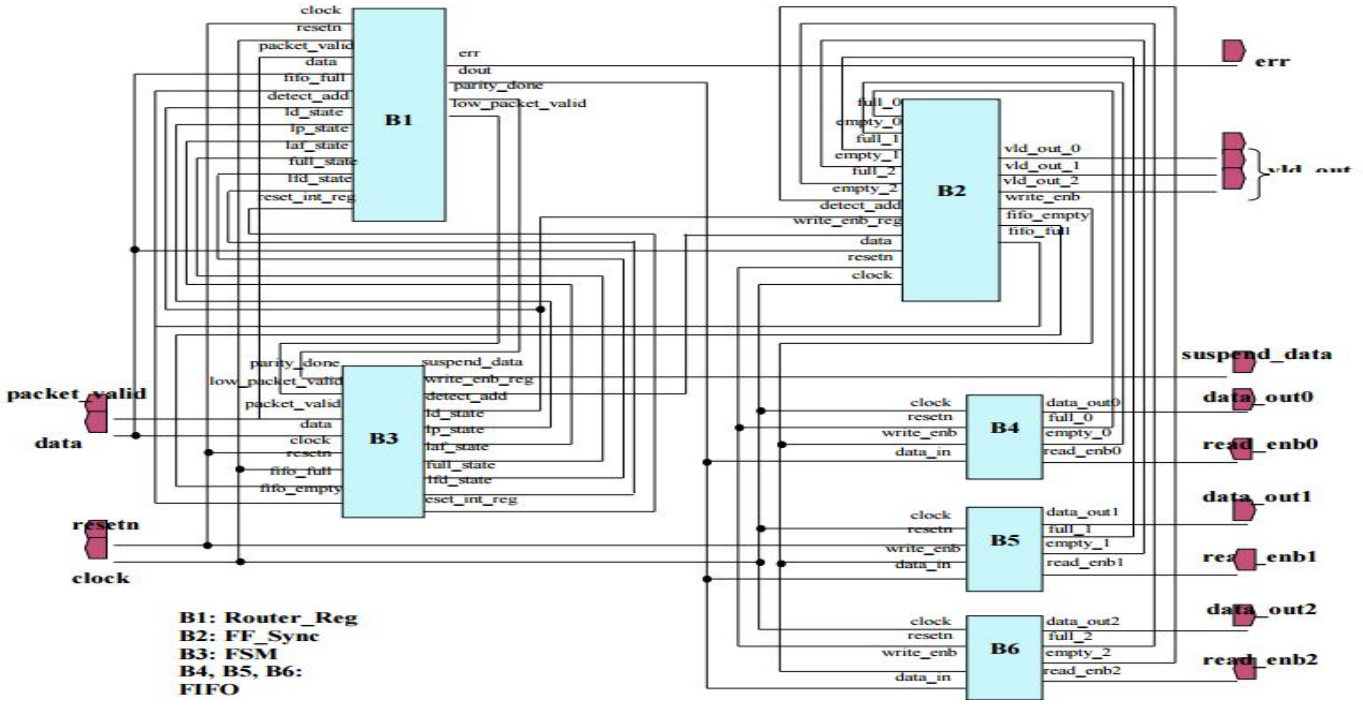
I. INTRODUCTION

A router is a networking device that forwards data packets between computer networks. Routers perform the traffic directing functions on the Internet. A data packet is typically forwarded from one router to another router through the networks that constitute an internetwork until it reaches its destination node.A router is connected to two or more data lines from different networks.When a data packet comes in on one of the lines, the router reads the network address information in the packet to determine the ultimate destination. Then, using information in its routing table or routing policy, it directs the packet to the next network on its journey. A router may have interfaces for different typesphysical layer connections, such as copper cables, fiber optic, or wireless transmission. Its firmware can also support different network layer transmission standards. Each network interface is used by this specialized computer software to enable data packets to be forwarded from one 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 network prefix. The network prefixes recorded in the routing table do not necessarily map directly to the physical interface connections.When multiple routers are used in interconnected networks, the routers can exchange information about destination addresses using a routing protocol. Each router builds up a routing table listing the preferred routes between any two systems on the interconnected networks.

II. ROUTER 1X3

Router consists of four main building blocks

A. Architecture of Router1x3



This design consists of 6 main blocks. Which are fsm_router, router_reg, ff_sync, and 3 fifo. The fsm_router block provides the control signals to the fifo, and router_reg module.

1) *Router_reg:* The router_reg module contains the status, data and parity registers for the router_1x3.These registers are latched to new status or input data through the control signals provided by the fsm_router.This module contains status, data and parity registers required by router.All the registers in this module are latched on rising edge of the ***clock***.Data registers latches the data from data input based on state and status control signals, and this latched data is sent thefifoor storage. Apart from it, data is also latched into the parity registers for parity calculation and it is compared with the parity byte of the packet. An ***error*** signal is generated if packet parity is not equal to the calculated parity.
2) *FF_sync:*Theff_sync module provides synchronization between fsm_router module and 3 fifo s , So that single input port can faithfully communicate with 3 output ports.This module provides synchronization between fsm and fifo modules. It provides faithful communication between single input port and three output ports. It will detect the address of channel and will latch it till packet_valid is asserted, address and write_enb_sel will be used for latching the incoming data into the fifo of that particular channel.Afifo_full output signal is generated, when the present fifo is full, and fifo_empty output signal is generated by the present fifo when it is empty.

   If data = 00 then fifo_empty = empty_0 and fifo_full = full_0
   If data = 01 then fifo_empty = empty_1 and fifo_full= full_1
   If data = 10 then fifo_empty = empty_2 and fifo_full = full_2
   Else fifo_empty = 0 and fifo_full = 1.
   The output vld_out signal is generated when empty of present fifo goes low, that means present fifo is ready to read.
   vld_out_0 = ~empty_0

vld_out_1 = ~empty_1vld_out_2 = ~empty_2The write_enb_reg signal which comes from th**e** fsm is used to generatewrite_enb signal for thepresentfifo which is selected by present address.

3) *FSM:*The 'fsm_router' module is the controller circuit for the router.This module generates all the control signals when new packet is sent to router. These control signals are used by other modules to send data at output, writing data into the fifo.
4) *FIFO Block:*There are 3 fifo for each output port, which stores the data coming from input port based on the control signals provided by fsm_routermodule.fresetn is low then full =0, empty = 1 and data_out = 0. Write operation: The data from input data_in is sampled at rising edge of the clock when input write_enb is high and fifo is not full. Read Operation: The data is read from output data_out at rising edge of the clock, when read_enb is high and fifo is not empty. ead and Write operation can be done simultaneously. Full – it indicates that all the locations inside fifo has been written. Empty – it indicates that all the locations of fifo are empty.

III. UNIVERSAL VERIFICATION METHODOLOGY

The UVM (Universal Verification Methodology) was introduced in December 2009, by a technical Sub committee of Accellera. UVM uses Open Verification Methodology as its foundation. Accellera released version UVM 1.0 EA on May 17, 2010.UVM Class Library provides the building blocks needed to quickly develop well-constructed and reusable verification components and test environments. It uses systemVerilog as its language. All three of the simulation vendors (Synopsys, Cadence and Mentor) support UVM today which was not the case with other verification methodology. Today, more and more logic is being integrated on the single chip so verification of it is a very challenging task. More than 70 percent of the time is spent on the verification of the chip. So it is a need of an hour to have a common verification methodology that provides the base classes and framework to construct robust and reusable verification environment. UVM provides that. In this paper, all the terminology related to UVM is introduced along with the sample example. In first phase uvm components are introduced. In second phase some of the features related to UVM are introduced and in final phase small environment is built using UVM from the scratch.

IV. UVM TESTBENCH ARCHITECTURE

The following subsections describe the components of a verification component.

A. Data Item (Transaction)

Data item are basically the input to the device under test. All the transfer done between different verification components in UVM is done through transaction object. Networking packets, instructions for processor are some examples of transactions. From the top level test many data items are generated and applied to the DUT so by intelligently randomizing the data items object we can check corner cases and Maximize the coverage on the device under test.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue X, October 2017- Available at www.ijraset.com

B. Driver (BFM)

Driver as the name suggest, drive the DUT signals. It basically receives the transaction object from the sequencer and converts it into the pin level activity. It is the active part of the verification logic.

C. Sequencer

Sequencer is the component on which the sequences will run. The DUT needs to be applied a sequence of transaction to test its behaviour. So sequence of transaction is generated and it is applied to driver whenever it demands by the sequencer.

D. Monitor

A monitor is the passive element of the verification environment. It just sample the DUT signal from the interface but does not drive them. It collect the pin information, package it in form of a packet and then transfer it to scoreboard or other components for coverage information.

E. Agent

Agent is basically a container. It contains driver, monitor and sequencer. Driver and sequencer are connected in agent. Agent has two modes of operation: passive and active. In active mode it drives the signal to the DUT. So driver and sequencer are instantiated in active mode. In passive mode it just sample the DUT signals does not drive them. So only monitor is instantiated in passive mode.

F. Scoreboard

Scoreboard is a verification component that checks the response from the DUT against the expected response. So it keeps track of how many times the response matched with the expected response and how many time it failed.

G. Environment

Environment is at the top of the test bench architecture, it will contain one or more agents depend on design. If more than one agents are there then it will be connected in this component. Agents are also connected to other components like scoreboard in this component.

V. RESULTS AND DISCUSSION

Following components of router were synthesized and simulated, using Xilinx ISE software. And Simulation results were observed.

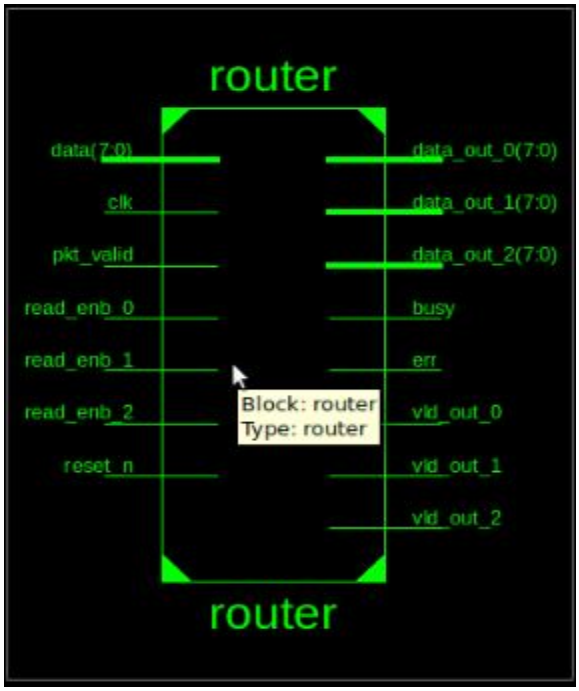


Fig 1:Router1X3 Schematic Diagram from Synthesis

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue X, October 2017- Available at www.ijraset.com

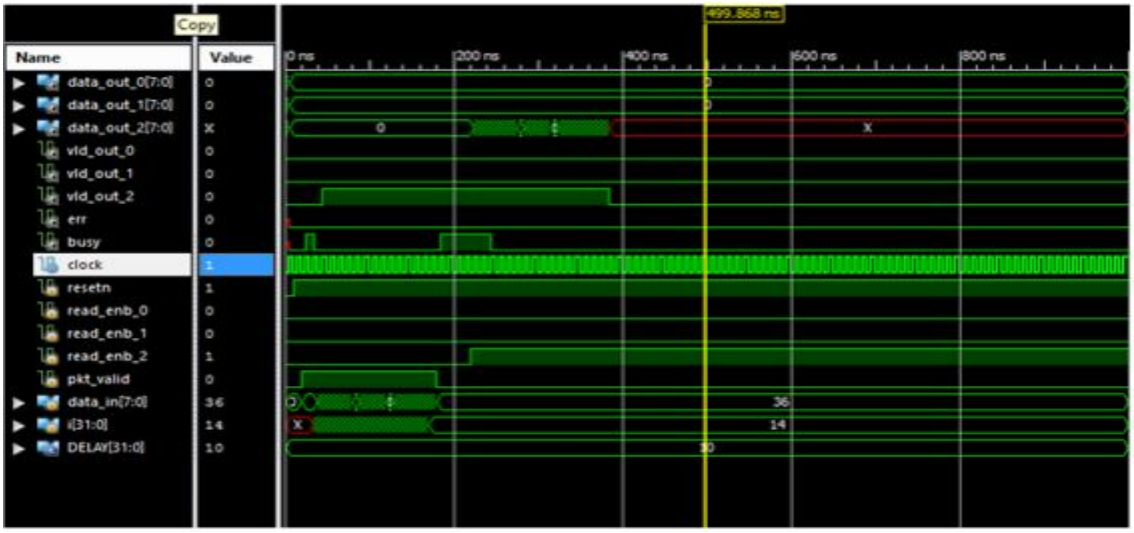


Fig2: Top module output of Router1X3

VI. COVERAGE REPORT

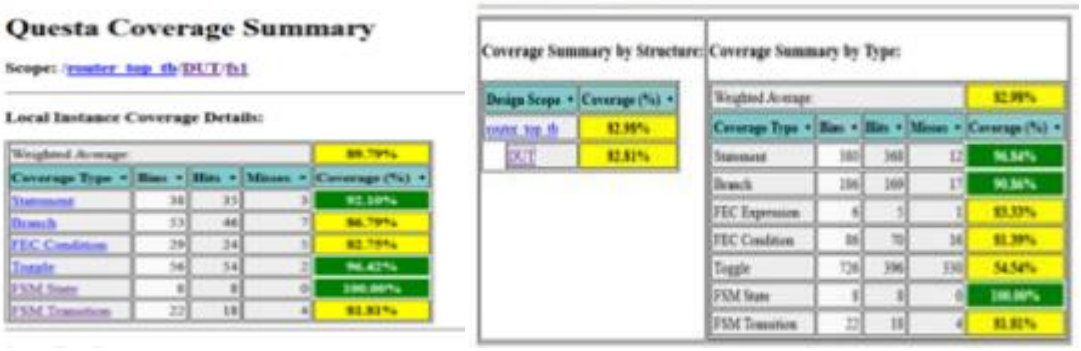


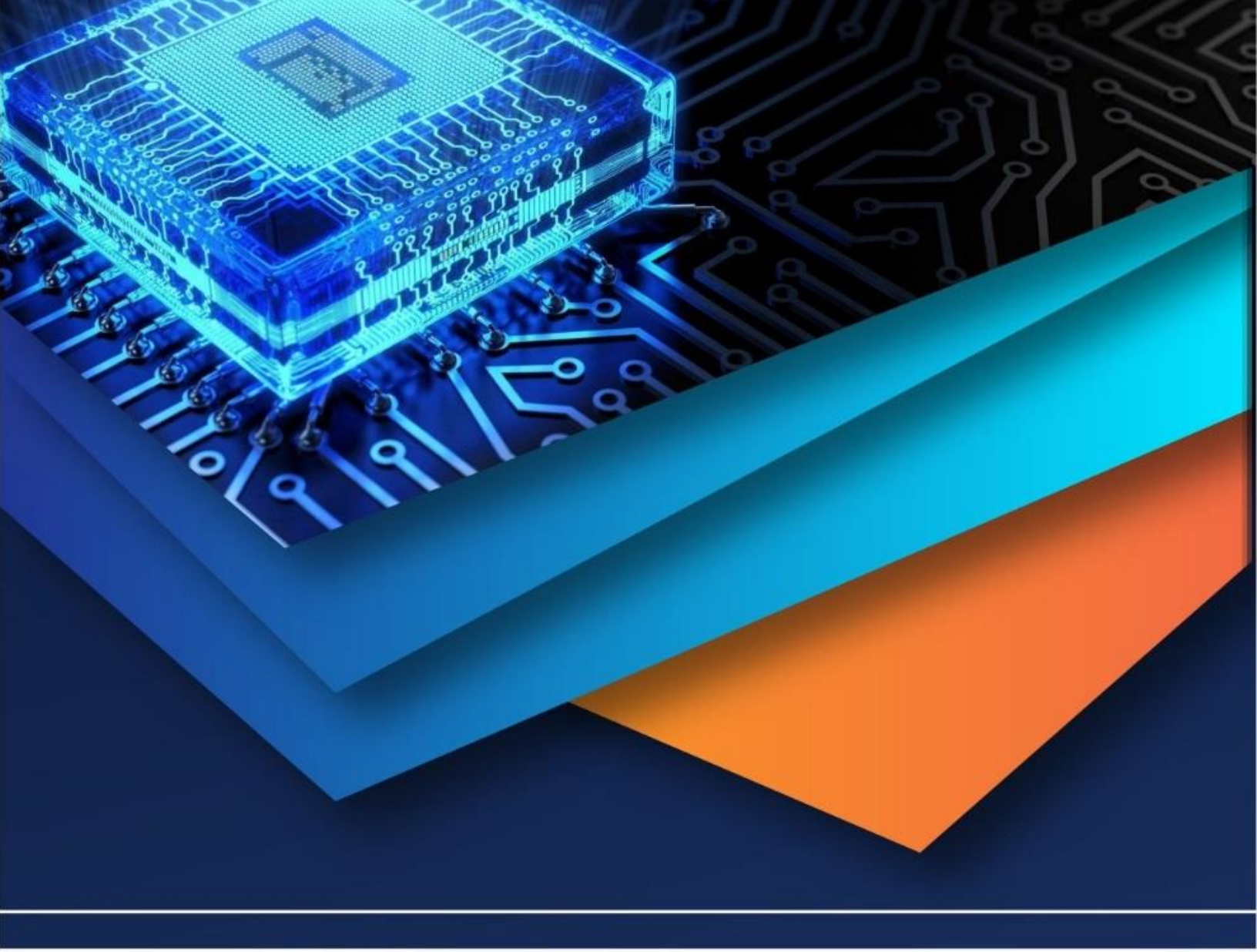
Fig 3 Functional code coverage of Router1X3

VII. CONCLUSION

In this we have designed and verified the Router1X3 core using Verilog and UVM technique using Questasim. Many coding bugs are debugged during the verification.The code coverage is obtaned for the RL design and 100% code coverage and functional coverage is extracted. This methodology provides the complete coverage of the RTL design of Router1X3.

REFERENCES

[1]. Verilog HDL:- Samir Palnitkar
[2]. System Verilog:- Chris Spear
[3]. The ASIC Website: www.acworld.com
[4]. www.google.com/fifo-theory-verilog
[5]. www.wikipedia.com
[6]https://en.wikipedia.org/wiki/Wireless_router
[7]opencores.org



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY