iJRASET
International Journal For Research in Applied Science and Engineering Technology

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 9 **Issue:** VII **Month of publication:** July 2021

**DOI:** https://doi.org/10.22214/ijraset.2021.36226

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: 7.429*
*Volume 9 Issue VII July 2021- Available at www.ijraset.com*

# Router1x3 Protocol Design Implementation and Verification with Virtual Cut through Mechanism for Network on Chip (NoC)

Shambhavi[1], Sumanth.S[2]
[1,2] *ECE Department (VLSI Design and Embedded systems), PES College of Engineering Mandya, Karnataka (India)*

***Abstract-Hundreds of processors and memory cores are implemented on a single substrate called the System on Chip (SoC). The SoC with bus-based architecture has restrictions on the processing speed of the system and as the design becomes complex and the issue of scalability arises. Hence NoC is designed to enhance the scalability, data reliability, and processing speed with low power consumption by decoupling communication from computations [1]. Using NoC the IP cores of SoC are connected through on-chip routers and send data to each other through packet switching. The router is a processing chip that decides the right path for data transmission, hence the efficient design of the router is essential to enhance the performance and throughput of the system [2]. To reduces latency through the switch, the Virtual cut-through mechanism is a packet switching technique, in which the switch starts forwarding a packet as soon as the destination address is processed by header.***

***Hence the present work focuses on a router input-output protocol design with the Virtual Cut-through mechanism for closed-loop communication. Router 1x3 has a single input port and three output ports. The architecture of Router 1x3 with sub-modules such as FIFO, FSM, Synchronizer, and Register is designed analyzed and verified using Verilog, System Verilog language, and Universal Verification Methodology(UVM). And it is also implemented on Xilinx 14.5 IDE with Spartan-6- XC6SLX45 FPGA.***

***Keywords: SoC, NoC, Router, FSM, FIFO, Synchronizer, Register***

## I. INTRODUCTION

Billions of transistors, millions of gates, thousands of circuits, and hundreds of cores on a single IC chip is the System on Chip (SoC). An efficient On-Chip that provides communication between these hundreds of IP cores is the NoC.

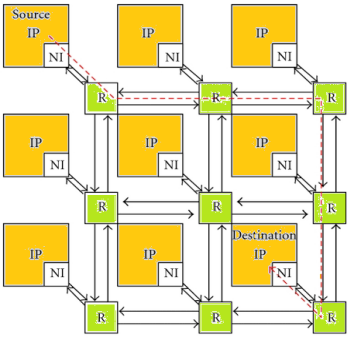



*Figure 1:NoC Architecture*

NoC's are implemented by Links which provides connections between IP cores, Routers are the processing elements responsible for routing information from a source port to its destination port for data communication and Network interfaces(NI) are logical connections between IP cores which separates the computation from communication. Since Router is a processing block of NoC, the efficient design of Router is more significant.

## II. VIRTUAL CUT-THROUGH(VTC)

To reduce packet store time in every switch, the VCT switching mechanism [3] was implemented by Kermani and Kleinrock. In this mechanism as soon as the header is processed to find the destination address, the packet flit cut-through into the next router if the outgoing channel is free.

Similarly, every flit is buffered in every router and immediately cut through to the next router if the output channel is free. In case of no resource conflicts along the path, the packet reaches its destination. Buffers are required when a busy channel is encountered.

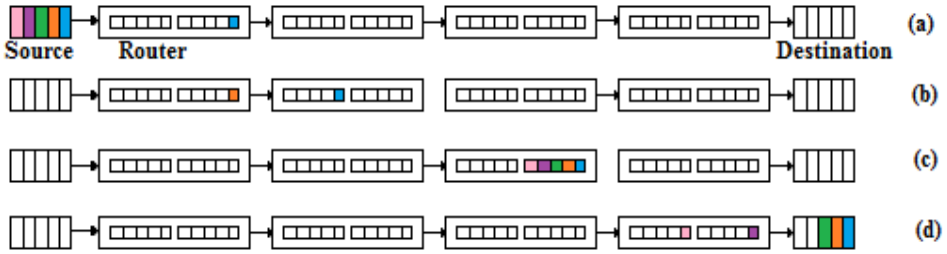



Figure: 4 Virtual cut-through mechanism

The figure shows the Virtual cut-through switching of a packet which consists of

- The header flit is processed and sent to the output FIFO of the first router.
- The header has cut through into the second router and all flits of a packet are following its path.
- The header has cut through into the next router and stored in the FIFO of that router due to a busy channel.
- Packet flit moving towards the destination.

## III. ARCHITECTURE OF ROUTER 1X3

The architecture of Router 1X3 consists of sub-modules such as FSM, REGISTER, SYNCHRONIZER, FIFO_0, FIFO_1, and FIFO_2. Each of these sub-modules is designed using Verilog RTL code, implemented on Spartan-6, and verified using System Verilog and UVM.

### *A. Router: FIFO*

Three FIFOs are used in the router design. Each FIFO has a width of 16 bits and a depth of 4 bits. The operation of FIFO is synchronized by the clock and an active low resetn signal is used to reset the FIFO. Soft_reset is an active high signal by the SYNCHRONIZER which resets the FIFO during a time-out state of the ROUTER. Full and empty are the outputs that indicate FIFO memory full state and no data in the FIFO state respectively.

A memory of size 16X4 bits is created to store the incoming packet whenever the channel is busy. The write_ptr and read_ptr signals are generated to point to the memory location during write and read operation respectively. Upon reset (resetn=0), full, empty, data_out, and memory are initialized to 0.

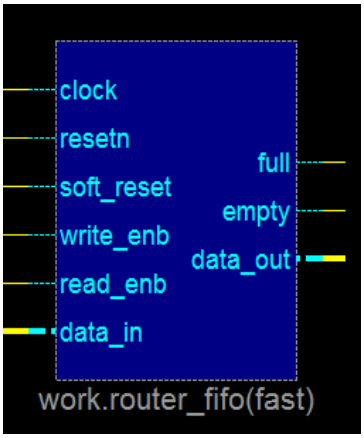



Figure 2:FIFO

*1) Write Operation:*

*a)* When write_enb is high and FIFO full is 0, the packet data_in is written to memory at the rising edge of the clock.

*b)* write_ptr is incremented to store the next packet to the memory.

*2) Read Operation:*

*a)* When read_enb is high and FIFO is not empty, the data packet is read from memory to data_out.
*b)* As soon as the data is read from a memory location, that location is initialized to 0.
*c)* The read_ptr is incremented by 1 to point to the next memory location.
*d)* The write_ptr is decremented by 1 to write next to the previously read memory location.
*e)* Read and write operation is done simultaneously.

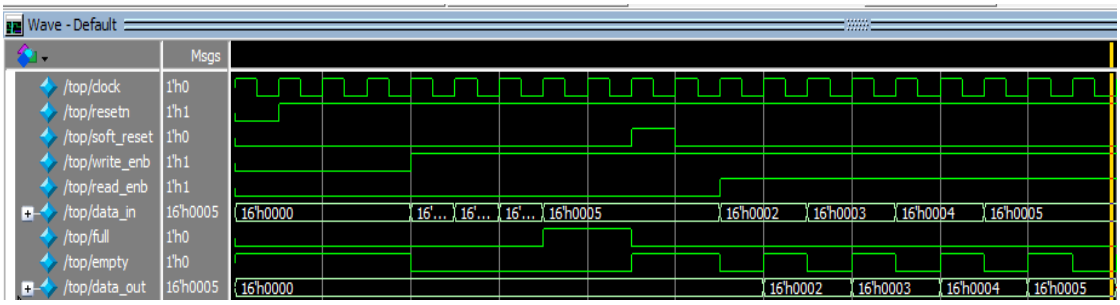



Figure 3:FIFO Waveform

*B. Router: Synchronizer*

This sub-block checks for the validity of data packets and reset router FSM and router FIFO by generating the soft_reset signal. And it also provides a write_enable signal to one of the FIFO.

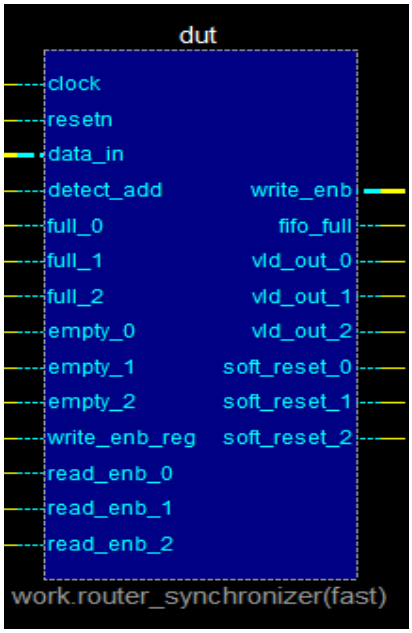



Figure 4:Synchronizer

In the router design fifo_full signal is asserted based on full_status of FIFO_0 or FIFO_1 or FIFO_2. These FIFO are selected based on the header bits of input data i.e if data_in =2'b00 then fifo_full=full_0; data_in=2'b01 then fifo_full=full_1; data_in=2'b10 then fifo_full=full_2 else fifo_full=0.

The vld_out_x signal is asserted based on empty signal from respective FIFO :

- vld_out_0=~empty_0
- vld_out_1=~empty_1
- vld_out_2=~empty_2

Hence when a data packet is sent to FIFO, respective vld_out_signal will be asserted and if those packets are not read from the destination, the synchronizer starts the counter. When the counter reaches 29 soft_reset signals are generated for that FIFO and FSM.

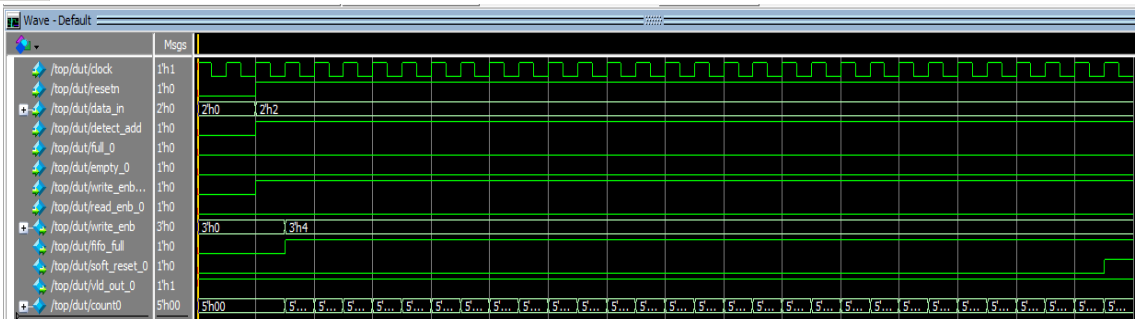



Figure 5:Synchronizer Waveform

*C. Finite State Machine(FSM)*

FSM is the monitoring and controlling unit of the router architecture. Moore's type of FSM is used in this design.

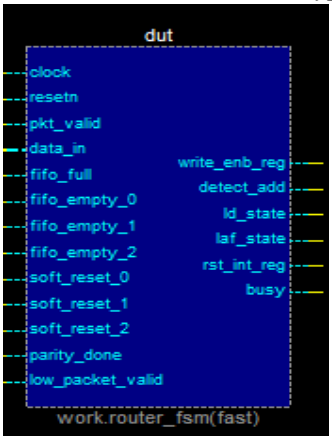



Figure 6:FSM

The router FSM consists of 6 states as follows:

DECODE_HEADER

Ø This is the initial state in which if the incoming packet is valid, the first two bits i.e the header bits are processed to select one of the FIFO.

If the selected FIFO is empty, transit to LOAD_DATA state else transit to WAIT_TILL_EMPTY state.

Ø Signal detect_add is asserted in this state.

LOAD_DATA

Ø The signal ld_state and write_enb_reg is asserted in this state.

Ø Signal busy is made low in this state, so that ROUTER can receive the new data from input source every clock cycle.

Ø This state transits to LAOD_PARITY state when pkt_valid goes low and to FIFO_FULL_STATE when FIFO is full.

FIFO_FULL_STATE

Ø Busy signal asserted and write_enb_reg signal is made low.

If FIFO is not full the control goes to LOAD_AFTER_FULL state else, it waits in this state.

LOAD_AFTER_FULL

Ø In this state laf_state, busy & write_enb_reg signal is asserted.

Ø In this state if the parity_done signal is high state changes to DECODE_HEADER else, it checks for low_packet_valid signal. and if the low_packet_valid is high, the control goes to PARITY state else it goes to LOAD_DATA state.

WAIT_TILL_EMPTY

Ø Busy signal is made high and write_enb_reg signal is made low.

This state waits until FIFO becomes empty and changes to LOAD_DATA state as soon as any of the FIFO becomes empty.

PARITY

Ø This state reset low_packet_valid signal by asserting rst_int_reg signal.

Ø The control changes to DECODE_ADDRESS when FIFO is not full and to FIFO_FULL_STATE when FIFO is full.

Ø Busy is made high in this state.

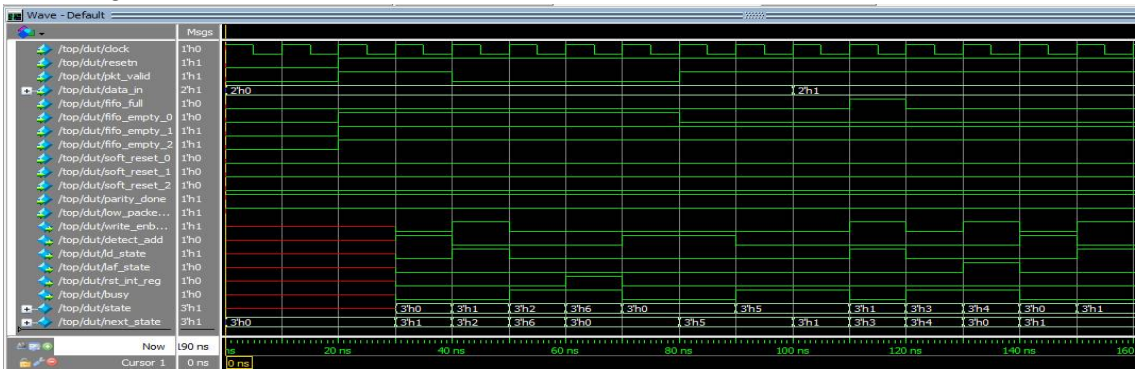



Figure 7:FSM Waveform

*D. ROUTER: Register*

This module implements an internal register called data_reg to hold the data packet when the respective FIFO is full. And as soon as the lfd_state signal from FSM asserts, the data in data_reg is sent to the respective FIFO.

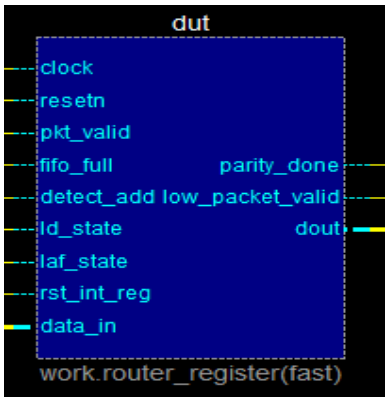



Figure 8:Register

When resetn is low then the signals dout, parity_done and low_pkt_valid are low. The signal parity_done is high, when ld_state is high and fifo_full and pkt_valid are low. If the pkt_valid signal is not asserted during the LOAD_DATA state low_packet_valid is made high indicating that the loading packet is not valid.

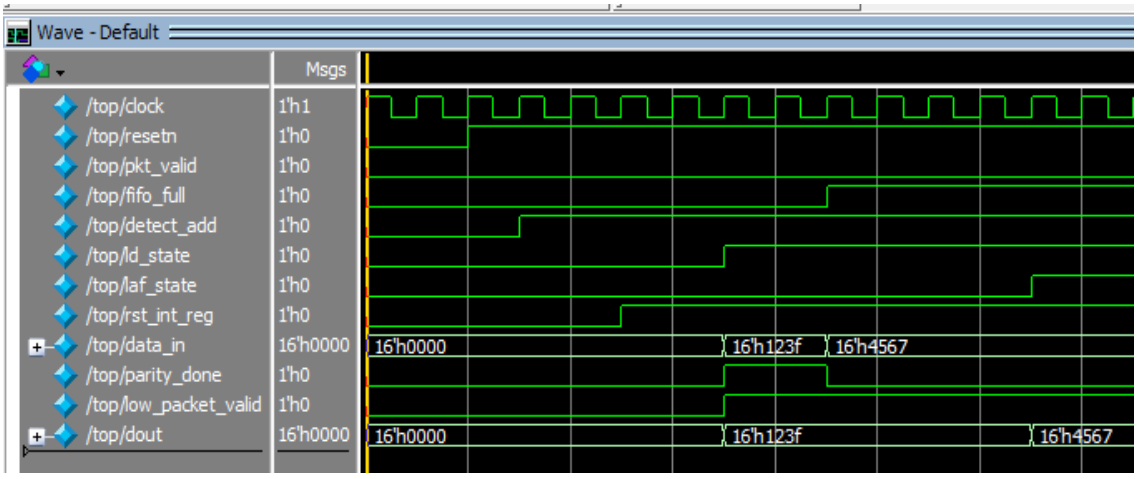



Figure 9:Register Waveform

*E. Router1x3*

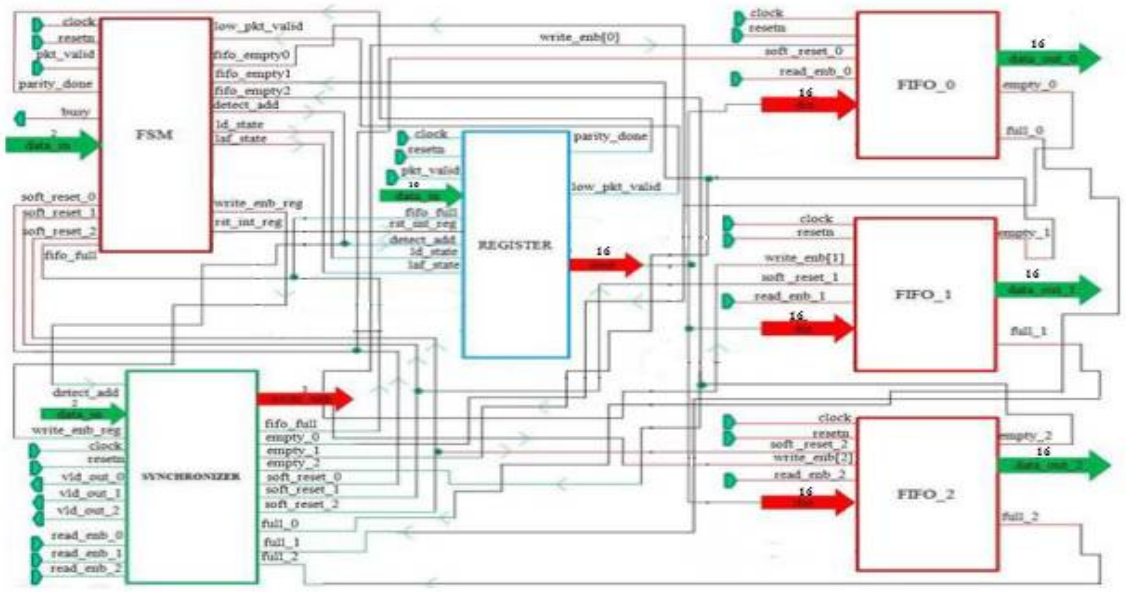



Figure 10:Router 1X3

The Router 1x3 consists of five input ports and seven output ports.

*a)* The input port datain is connected to FSM, Synchronizer, and Register, and the first two bits of the input are used by FSM and Synchronizer to select one of the FIFO i.e {00} for FIFO_0, {01} for FIFO_1 and {10} for FIFO_2.

*b)* The input port pkt_valid is connected to Register and FSM to indicate the incoming packet is valid or not.

*c)* The input ports read_enb_0, read_enb_1 and read_enb_2 are connected to FIFO_0, FIFO_1 and FIFO_2 respectively to initiate read operation.

*d)* The output ports data_out_0, data_out_1 and data_out_2 from FIFO_0, FIFO_1 and FIFO_2 respectively are used send data to destination.

*e)* The output ports vld_out_0, vld_out_1, and vld_out_2 from the synchronizer assert when valid data transmitted to the respective FIFO.

*f)* The output port busy from FSM specifies that the header is decoded and until the pkt_valid is high the all the data is sent to the respective FIFO.

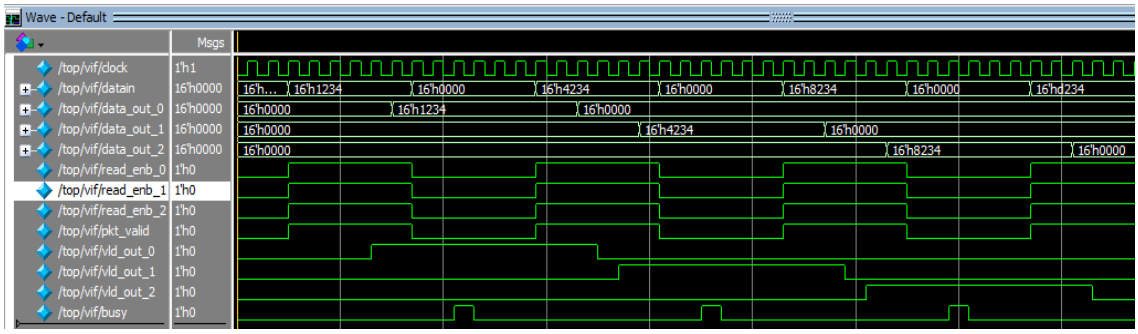



Figure 11:Router 1X3 Waveform

## IV. IMPLEMENTATION OF SPARTAN-6

The Spartan-6 FPGA XC6SLX45 CSG324C is a high-capacity device used to implement most of the digital systems. Featuring high-speed DDR2 memory, multiple HDMI ports, Gigabit Ethernet, and advanced clocking and power supply circuits. The Router1X3 design is implemented on Spartan-6 FPGA using XILINX and verified the output using Chip Scope-Pro.

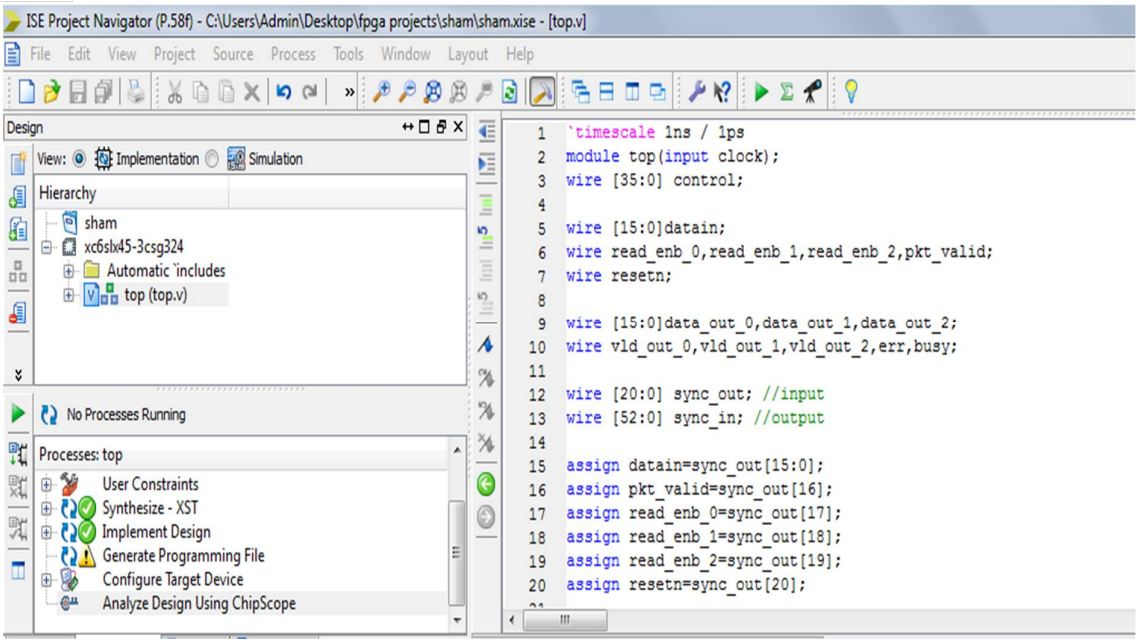



Figure 12:Synthesis and design implementation in XILINX

| Device Utilization Summary | | | |
|---|---|---|---|
| **Slice Logic Utilization** | **Used** | **Available** | **Utilization** |
| Number of Slice Registers | 1,169 | 54,576 | 2% |
| Number used as Flip Flops | 1,169 | | |
| Number used as Latches | 0 | | |
| Number used as Latch-thrus | 0 | | |
| Number used as AND/OR logics | 0 | | |
| Number of Slice LUTs | 428 | 27,288 | 1% |
| Number used as logic | 319 | 27,288 | 1% |
| Number using O6 output only | 143 | | |
| Number using O5 output only | 15 | | |
| Number using O5 and O6 | 161 | | |
| Number used as ROM | 0 | | |
| Number used as Memory | 21 | 6,408 | 1% |

Figure 13:Design utilization Summary of Router1X3

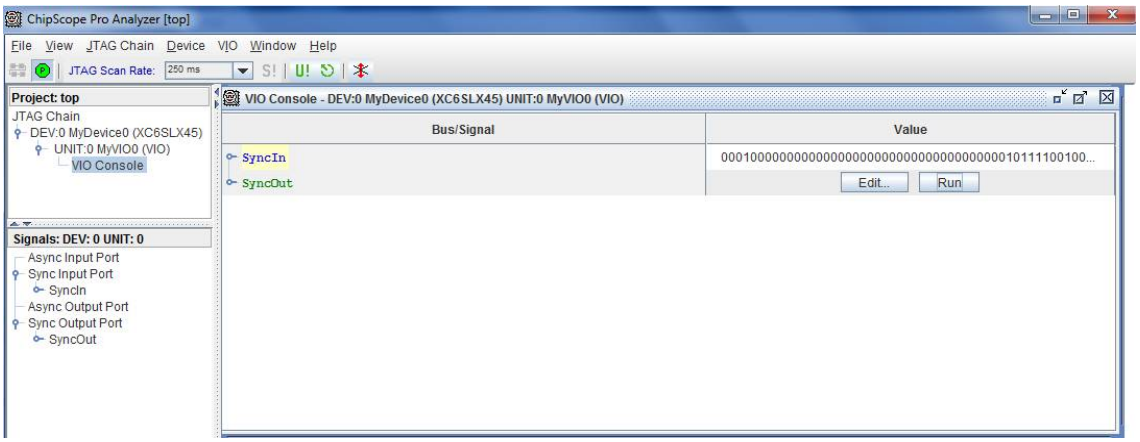



Figure 14:REsult on ChipScope Pro for data_out_0

## V. VERIFICATION USING SYSTEM VERILOG AND UVM

The above design is verified using System Verilog by implementing the environment components such as generator, BFM (Bus Functional Model), interface, monitor, scoreboard, coverage, and assertion to achieve scoreboard, functional coverage, code coverage, and assertion. The Universal Verification Methodology(UVM) is an advanced methodology, widely used in industries because of its vast reusable library classes. UVM verification environment consists of Sequencer, driver, agent, interface, monitor, scoreboard, assertion, coverage, and test classes to verify the Router1X3 design. Most of these classes inherit uvm_component, uvm_object, and uvm_sequence.

```
# datain=4234
# read_enb_0=0
# read_enb_1=1
# read_enb_2=1
# pkt_valid=1
# data_out_0=0000
# data_out_1=4234
# data_out_2=0000
# vld_out_0=0
# vld_out_1=1
# vld_out_2=0
# busy=0
# SCOREBOARD PASS:::::
# :: ASSERTION PASS ::
```

Figure 15:SV Scoreboard Result for single input

```
# :: ASSERTION PASS ::
# ----------------------------------------------------------------------------
# Name                              Type        Size  Value
# ----------------------------------------------------------------------------
# req                               router_tx   -     @805
#   datain                          integral    16    'h1234
#   read_enb_0                      integral    1     'h1
#   read_enb_1                      integral    1     'h0
#   read_enb_2                      integral    1     'h0
#   pkt_valid                       integral    1     'h1
#   vld_out_0                       integral    1     'h1
#   vld_out_1                       integral    1     'h0
#   vld_out_2                       integral    1     'h0
#   data_out_0                      integral    16    'h1234
#   data_out_1                      integral    16    'h0
#   data_out_2                      integral    16    'h0
#   begin_time                      time        64    25
#   depth                           int         32    'd2
#   parent sequence (name)          string      3     seq
#   parent sequence (full name)     string      30    uvm_test_top.env.agent.sqr.seq
#   sequencer                       string      26    uvm_test_top.env.agent.sqr
# ----------------------------------------------------------------------------
# SCOREBOARD PASS:::::
```

Figure 16:UVM Scoreboard and Assertion Report of single input

Covergroups

| Name | Class Type | Coverage | Goal | % of Goal | Status | Included | Merge_instances | Get_inst_coverage | Comment | % over Goal | Covered Bins (Hits) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| /run_svh_unit/rout... | | | | | | | | | | | |
| TYPE router_c... | router_cov | 100.0% | 100 | 100.0% | ✓ | | auto(1) | | | 100.0% | 14 |
| CVP router... | router_cov | 100.0% | 100 | 100.0% | ✓ | | | | | 100.0% | 2 |
| CVP router... | router_cov | 100.0% | 100 | 100.0% | ✓ | | | | | 100.0% | 2 |
| CVP router... | router_cov | 100.0% | 100 | 100.0% | ✓ | | | | | 100.0% | 2 |
| CVP router... | router_cov | 100.0% | 100 | 100.0% | ✓ | | | | | 100.0% | 2 |
| CVP router... | router_cov | 100.0% | 100 | 100.0% | ✓ | | | | | 100.0% | 2 |
| CVP router... | router_cov | 100.0% | 100 | 100.0% | ✓ | | | | | 100.0% | 2 |
| CVP router... | router_cov | 100.0% | 100 | 100.0% | ✓ | | | | | 100.0% | 1 |
| CVP router... | router_cov | 100.0% | 100 | 100.0% | ✓ | | | | | 100.0% | 1 |

Figure 17:SV and UVM Functional Coverage Report

Coverage Report Totals BY FILES: Number of Files 14

| Enabled Coverage | Active | Hits | Misses | Weight | % Covered |
|---|---|---|---|---|---|
| Stmts | 274 | 271 | 3 | 1 | 98.9 |
| Branches | 103 | 96 | 7 | 1 | 93.2 |
| Conditions | | | | 1 | 88.4 |
| UDP Condition Rows | 0 | 0 | 0 | 1 | 100.0 |
| FEC Condition Terms | 65 | 50 | 15 | 1 | 76.9 |
| Expressions | | | | 1 | 93.7 |
| UDP Expression Rows | 0 | 0 | 0 | 1 | 100.0 |
| FEC Expression Terms | 8 | 7 | 1 | 1 | 87.5 |
| FSMs | | | | 1 | 86.1 |
| States | 6 | 6 | 0 | 1 | 100.0 |
| Transitions | 18 | 13 | 5 | 1 | 72.2 |
| Toggle Bins | 818 | 432 | 386 | 1 | 52.8 |

Total coverage (Code Coverage Only, filtered view): 82.5%

Figure 18:SV and UVM Code Coverage Report

Assertions

| Name | Assertion Type | Language | Enable | Failure Count | Pass Count | Included | Design Unit | Design Unit Type |
|---|---|---|---|---|---|---|---|---|
| /top/ttu/A1 | Immediate | SVA | on | 0 | 1 | ✓ | assertion | Verilog |
| /top/ttu/A2 | Immediate | SVA | on | 0 | 1 | ✓ | assertion | Verilog |
| /top/ttu/A3 | Immediate | SVA | on | 0 | 1 | ✓ | assertion | Verilog |

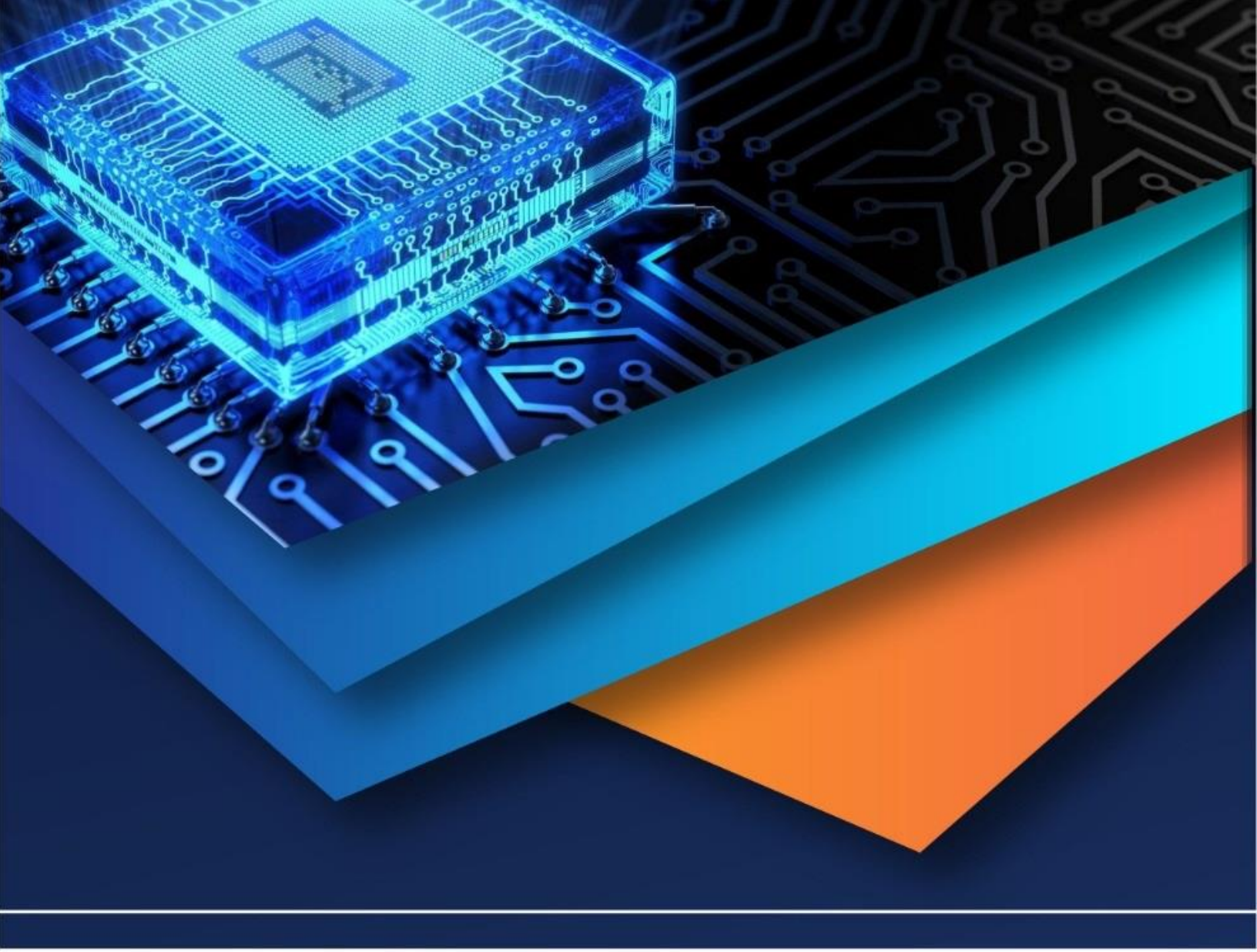
Figure 19: SV and UVM Assertion Report

## VI. CONCLUSION

The design using Verilog, implementation on Spartan-6, and verification using System Verilog and UVM of the Router1X3 are achieved with 100% functional coverage, 82.5% Code coverage, 100% immediate assertion, scoreboard, and waveform with all possible conditions.

## REFERENCES

[1] D. Naresh Kumar and Geethu Mohan, "Design and Optimization of System-on-chip (SOC)", Journal of Emerging Technologies and Innovative Research (JETIR),2015.
[2] Vikrant A. Bute and Devendra S. Chaudhari, "Review on Network on Chip Router Design" International Journal of Computer Science and Information Technologies,2004.
[3] Pongstorn Maidee and Alireza Kaviani, "Improving FPGA NoC Performance using Virtual Cut-Through Switching Technique " IEEE conference, 2005.
[4] Nidhi Gopal, "Router 1X3 – RTL Design and Verification", International Journal of Engineering Research and Development
[5] Priyanka Rakshe and Pankaj Prajapati," "Router1x3 Protocol design with Virtual cut through Mechanism for Network on Chip (NoC)", Journal of Emerging Technologies and Innovative Research
[6] Ramya. C.B, Chethan. C.M, Praveen. S.G, Rakshith S, "Design and Implementation of Spartan 6 Series FPGA Board and Application of PID Controller for Robots", International Journal of Recent Advances in Engineering & Technology (IJRAET) Volume-3, Issue -5, 2015



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY