Functional Verification of Logic Modules for a Ethernet Switch

Venkat Pedada[1], Namala Sreeja[2], Bheemaraju Jalandhar[3], Tulasi Naveen Kumar[4], Mrs.K. Deepa Rao[5], Mr.D. Srikanth[6]

[1,2,3,4] *Department of Electronics and Communication Engineering, J.B. Institute of Engineering & Technology (UGC Autonomous, Permanently Affiliated to Jawaharlal Nehru Technological University, Hyderabad) Bhaskar Nagar (Post), Moinabad Mandal, R.R. Dist.-500075*

[5,6] *Asst. Professor, Department of Electronics and Communication Engineering, J.B. Institute of Engineering & Technology (UGC Autonomous, Permanently Affiliated to Jawaharlal Nehru Technological University, Hyderabad) Bhaskar Nagar (Post), Moinabad Mandal, R.R. Dist.-500075*

***Abstract*— This work presents the functional verification of logic modules for a Ethernet Switch for an ASIC based on the NetFPGA platform. A coverage-driven constrained random stimulus approach is used. It is implemented in a layered testbench environment with self-checking capability. This environment implements the methodology presented by the Verification Methodology Manual (VMM) using SystemVerilog. The main advantage of this methodology is its reusability. This characteristic enables the development of a common testbench environment for our modules with minimum changes for each particular module. The four logic modules presented in this work implement functions of a Ethernet switch. The common characteristic of these circuits is the close dependency between the time and its functionality. These modules need time information to deal with problems such as rate limiting, quality of service (QoS) or aging lookup tables in classification engines. As described in the literature, the transaction-level models used to predict the circuit behavior are time-independent when the implementation details are not relevant. But when time information influences the circuit functionality, the model needs to replicate the circuit latency to be functionally equivalent. We propose a simple solution to the synchronization process between the model and the design under verification (DUV). This solution preserves the main advantage of transaction-level models (faster simulation time than the RTL model) and generates the result data with the same circuit latency. These features made possible to run a considerable amount of test cases that helps to find and correct bugs in the circuit with a high confidence measured by the functional and code coverage results.**

***Index Terms:* VMM, Packets, Scoreboard, RTL etc.**

I.INTRODUCTION

In this paper, we will verify the Switch RTL core. Following are the steps we follow to verify the Switch RTL core.

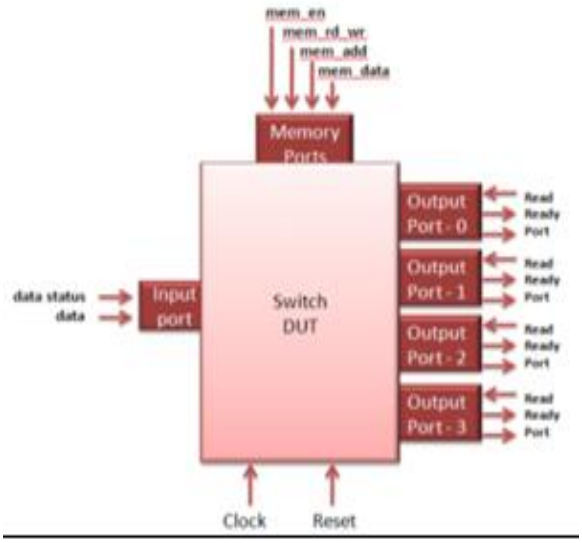
1) Understand the specification
2) Developing Verification Plan
3) Building the Verification Environment. We will build the Environment in Multiple phases, so it will be easy for you to lean step by step.

Phase 1) We will develop the testcase and interfaces, and integrate them in these with the DUT in top module.
Phase 2) We will Develop the Environment class.
Phase 3) We will develop reset and configuration methods in Environment class. Then using these methods, we will reset the DUT and configure the port address.
Phase 4) We will develop a packet class based on the stimulus plan. We will also write a small code to test the packet class implementation.
Phase 5) We will develop a driver class. Packets are generated and sent to dut using driver.
Phase 6) We will develop receiver class. Receiver collects the packets coming from the output port of the DUT.
Phase 7) We will develop scoreboard class which does the comparison of the expected packet with the actual packet received from the DUT.

Phase 8) We will develop coverage class based on the coverage plan.
Phase 9) In this phase , we will write test cases and analyze the coverage report.

II. SPECIFICATION:

SWITCH SPECIFICATION:
This is a simple switch. Switch is a packet based protocol. Switch drives the incoming packet which comes from the input port to output ports based on the address contained in the packet.
The switch has a one input port from which the packet enters. It has four output ports where the packet is driven out.



Packet Format:
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 4 bytes to 259 bytes.
Packet Header:
Packet header contains three fields DA, SA and length.

DA: Destination address 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.

SA: Source address of the packet from where it originate. It is 8 bits.

Length: Length of the data is of 8 bits and from 0 to 255. Length is measured in terms of bytes.
If Length = 0, it means data length is 0 bytes
If Length = 1, it means data length is 1 bytes
If Length = 2, it means data length is 2 bytes
If Length = 255, it means data length is 255 bytes

Data: Data should be in terms of bytes and can take anything.

FCS: Frame check sequence
This field contains the security check of the packet. It is calculated over the header and data.

Configuration:
Switch has four output ports. These output ports address have to be configured to a unique address. Switch matches the DA field of the packet with this configured port address and sends the packet on to that port. Switch contains a memory. This memory has 4 locations, each can store 8 bits. To configure the switch port address, memory write operation has to be done using memory interface. Memory address (0,1,2,3) contains the address of port(0,1,2,3) respectively.

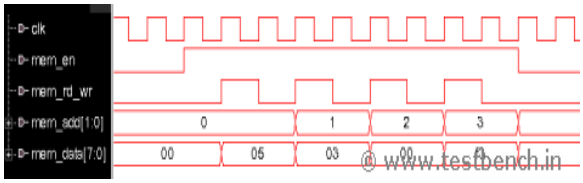
Interface Specification:

The Switch has one input Interface, from where the packet enters and 4 output interfaces from where the packet comes out and one memory interface, through the port address can be configured. Switch also has a clock and asynchronous reset signal.

Memory Interface:
Through memory interfaced output port address are configured. It accepts 8 bit data to be written to memory. It has 8 bit address inputs. Address 0,1,2,3 contains the address of the port 0,1,2,3 respectively.

There are 4 input signals to memory interface. They are
input mem_en;
input mem_rd_wr;
input [1:0] mem_add;
input [7:0] mem_data;

All the signals are active high and are synchronous to the positive edge of clock signal.
To configure a port address,
1. Assert the mem_en signal.
2. Asser the mem_rd_wr signal.
3. Drive the port number (0 or 1 or 2 or 3) on the mem_add signal
4. Drive the 8 bit port address on to mem_data signal.



III.VERIFICATION PLAN

Overview
This Document describes the Verification Plan for Switch. The Verification Plan is based on System Verilog Hardware Verification Language. The methodology used for Verification is Constraint random coverage driven verification.
Feature Extraction:
This section contains list of all the features to be verified.
1) ID: Configuration
Description: Configure all the 4 port address with unique values.

2) ID: Packet DA
Description: DA field of packet should be any of the port address. All the 4 port address should be used.

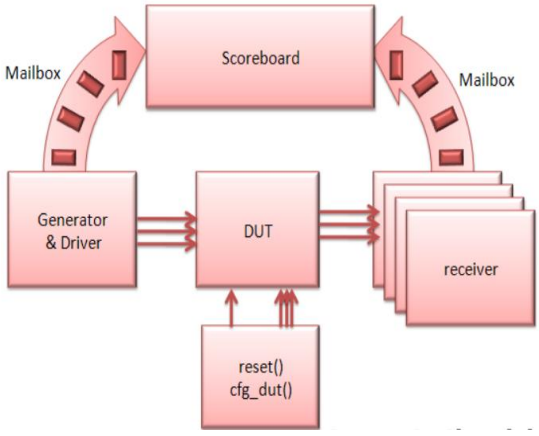
3)ID : Packet payload
Description: Length can be from 0 to 255. Send packets with all the lengths.

4) ID: Length
Description:
Length field contains length of the payload.
Send Packet with correct length field and incorrect length fields.

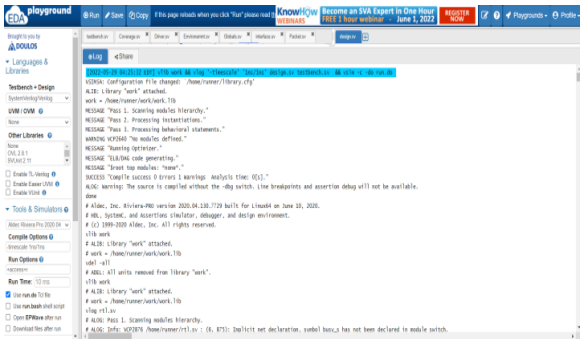
5) ID: FCS
Description:
Good FCS: Send packet with good FCS.
Bad FCS: Send packet with corrupted FCS.

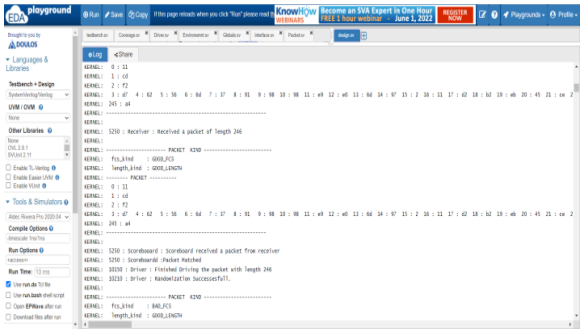
Stimulation Generation Plan:
1) Packet DA: Generate packet DA with the configured address.
2) Payload length: generate payload length ranging from 2 to 255.
3) Correct or Incorrect Length field.
4) Generate good and bad FCS.
Coverage Plan:
1) Cover all the port address configurations.
2) Cover all the packet lengths.
3) Cover all correct and incorrect length fields.
4) Cover good and bad FCS.
5) Cover all the above combinations.

IV.VERIFICATION ENVIRONMENT



V.VERIFICATION RESULTS





VI.ACKNOWLEDGEMENT

We are deeply grateful to Mrs.k.Deepa Rao, Mr.D.Srikanth who gave us guidance and suggestions and helped us in the successful completion of this project.

REFERENCE

[1] C. Spear, SystemVerilog for Verification, Second Edition: A Guide to Learning the Testbench LanguageFeatures. Springer Publishing Company, Incorporated, 2008

[2] M. Strum, W. J. Chau, and E. Romero, “Comparing two testbench methods for hierarchical functional verification of a bluetooth baseband adaptor,” in Hardware/Software Codesign and System Synthesis,2005. CODES+ISSS ’05. Third IEEE/ACM/IFIP International Conference on, 2005, pp. 327 –332.

[3] S. Vasudevan, Effective Functional Verification. Springer, 2006.

[4] M. Shreedhar and G. Varghese, “Efficient fair queuing using deficit round-robin,” Networking, IEEE/ACM Transactions on, vol. 4, no. 3, pp. 375 –385, Jun. 1996.

[5] M.-K. You, Y.-J. Oh, and G.-Y. Song, “Implementation of a hardware functional verification system using systemc infrastructure,” in TENCON 2009 - 2009 IEEE Region 10 Conference, 2009, pp. 1 –5.

[6] J. Bergeron, Writing Testbenches using SystemVerilog. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.