International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Generation of Graph for Ethernet Verification Using TREK

Vinodhini.M[1], Nambi.J.U[2], Kanmani.V[3]
[1,3]*Department of Electronics and Communication Engineering*
KPR Institute of Engineering and Technology,Arasur,Coimbatore.

Abstract— The main objective is Verification of ETHERNET PARSER AND ROUTER using Graph based scenario model. This includes generation of graph and integration of testcases. Graph generation is done using the software Trek. This Software automatically generates testcases which are self-verifying. The testcases generated from graph-based Scenario Model captures intended system behaviour. Trek takes its input information from scenario models describing the desired outcomes, developed by the user. This graph based scenario model is considered to be one of the most efficient method since test maintenance and test debugging can be carried out easily and also the test simulation run time required is less. Ethernet parser and address look up engine connects to ingress interface on one side and egress interface on the other. The Ethernet packet will be received into the switch through one of the input ports of ingress logic. Parser logic analyzes the packet contents and extracts the key fields. The Ethernet packet will be sent out from the switch through one of the output ports of egress logic. The full packet will be transferred in 10 clock cycles without any modification. Random input is given and graph is generated.
Keywords— Device Under Test (DUT) , Silicon On Chip (SOC) , Design Under Verification (DUV) , Equipment Under Test (EUT) .

I. INTRODUCTION

Today, Ethernet is the most popular LAN technology. It is an easy, relatively inexpensive way to provide high-performance networking to all different types of computer equipment. Ethernet has 10 Mbps throughput and uses CSMA/CD method to access the physical media. It allows devices and equipment to be accessed remotely and provides a cost-effective and reliable means of monitoring or controlling such equipment. It has advantage of transmitting a packet in bytes at a time. Graph based verification of Ethernet parser and router is done using Trek software. Graph is generated using this software. It generates testcases automatically. Trek product automatically generates multi-threaded, multi-processor, self-verifying C test cases for the SoC. This testcases are generated from graph-based scenario models that capture intended system behavior. The testcases generated from graph is given as input and Verilog coding is used to integrate with the testbench. It is then integrated with Device Under Test (DUT) and hence verification is done. The scenario model provides insight into the SoC design and the verification space that must be covered before the chip is fabricated.

II. VERIFICATION

Verification is the process used to demonstrate the functional correctness of the design. Verification process is used to make sure to implement what we want and to ensure the result of transformation is as expected one. It shows the presence of error. Verification environment is commonly referred as testbench. Testbench is the verification environment containing the set of components and interconnection of such components with Design Under Verification (DUV). Verification plan enables developing the testbench environment early. The verification team must understand all of the data flows and all possible interactions if it is to develop a testbench environment.

III.GRAPH MODEL

The graph defines the valid test space. Test space means providing solution for the testing device. The graphs are simpler to develop and maintain, and easier to reuse or acquire from third parties. Because a graph can be expressed as a diagram, it can be significantly easier to understand the stimulus, particularly for team-members who are not familiar with verification. Graph based verification gives benefits such as predictable coverage closure, facilitate reuse and productivity improvements. Graph algorithms enable fast, efficient coverage closure and test scenario generation. It can be easily integrate into existing environment. The graph is

©IJRASET 2015: All Rights are Reserved

Volume 3 Issue IX, September 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

generated using Trek. Random input is given and graph is generated. TrekSoc generates testcases automatically from the scenario model. Scenario model simplifies the generation of testcases. Verilog coding is used to integrate Graph with the testbench. It is then integrated with the DUT and hence verification is done. Trek usage model is effective and speeds up C test creation. Testcases are optimized for runtime efficiency. Trek is multithreaded, multiprocessor testcases and unlimited number of testcases can be tested. Testcase exercise end to end use cases. It target testcases for multiple platforms. In graph based verification same model can be reused between project and it also improves productivity.

IV. DEVICE UNDER TEST

Device Under Test (DUT) also known as equipment under test (EUT) and unit under test (UUT), is a term commonly used to refer to a manufactured product undergoing testing. A DUT is a device that is tested to determine performance and proficiency. A DUT is checked for defects to make sure the device is working.

A. Ethernet Parser And Router

Ethernet parser and address lookup engine consists of ingress interface on one side and egress interface on the other side. Parser logic analyzes the packet contents and extracts the key fields. A parser is a program usually part of a compiler that receives input in the form of sequential source program instructions that can be managed by other program. It is a compiler or interpreter component that breaks data into smaller elements for easy translation into another language. A parser takes input in the form of a sequence of tokens or program instructions and usually builds a data structure in the form of a parse tree or an abstract syntax tree. Ethernet parser accepts 64 bit data interface through ingress interface and same data is given as output without any modification through egress interface. It contains up to 32 input ports and output ports, for simplicity 4 ports are used. The packet size is fixed to 80 bytes and the full packet is transferred in 10 clock cycles. Only restriction is at any given point only one input port can send to one output port. External clock signal is given and reset option is used for resetting at active high.

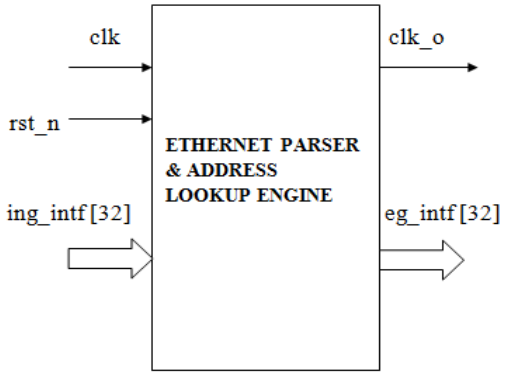


Fig. 1 Block diagram of Ethernet Parser and Router

B. Ingress Interface

Interface is referred as port. Ingress interface also referred as source. Ingress is the act of entering something. Ingress is a packet coming inside the interface. A packet arrived at the ingress interface is transported to the egress interface. A packet is a symbol or a sequence of 0's and 1's. An ingress interface is connected to an ingress line. An ingress line is said to feed the ingress interface. The Ethernet packet will be received into switch through one of the input ports**.** Each ingress port has data and data valid signal. It has 4 interface ports. In_port_data accepts data in 64 bits at a time. In_port_data_valid enables the port_in_data, it goes high at the start of the packet and remains high till end of packet reception. Input port_data_valid is a level signal remains high for 10 continuous clocks. One full packet gets transferred in 10 clocks.

©IJRASET 2015: All Rights are Reserved

Volume 3 Issue IX, September 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

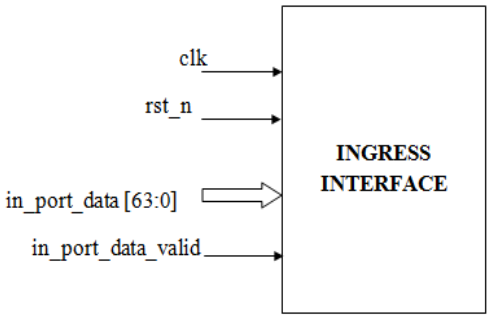


Fig. 2 Ingress Interface

TABLE I
SIGNAL DESCRIPTION FOR INGRESS INTERFACE

Signal	Width	Direction	Description
in_port_data	64	Input	It accepts the data in 64 bits at a time from Ethernet packet.
in_port_data_valid	1	Input	It enables the"port_in_data ".

C. Egress Interface

Egress interface also referred as sink. An egress interface is connected to an egress line. An egress line feeds an egress interface. Egress is the packet exists through the interface. Each Egress port has data, data_sop, data_eop data_status signals. The Ethernet packet will be sent out from the switch through one of the output ports. Out_port_data delivers the data from the switch in 64 bits at a time. Port_out_valid indicates the valid data from the switch.out_port_data_sop goes high at start of the packet and goes low in the next clock. For next 10 continuous clocks (including the current one) packet is sent out via out_port_data signal. out_port_pkt_status indicates the status of packet, is valid only during the "sop" cycle. On the last chunk, out_port_data_eop goes high for 1 clock.

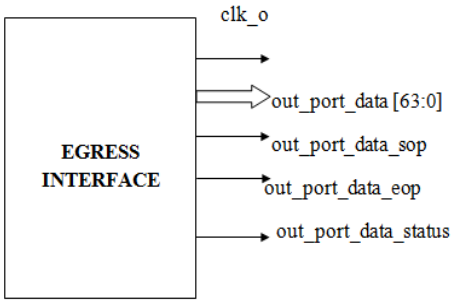


Fig. 3 Egress Interface

©IJRASET 2015: All Rights are Reserved

Volume 3 Issue IX, September 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

TABLE III
SIGNAL DESCRIPTION FOR EGRESS INTERFACE

Signal	Width	Direction	Description
out_port_data	64	Output	It delivers the data from the switch in 64 bits at a time.
out_port_data_sop	1	Output	It indicates the Start-Of-Packet for this output port from the switch.
out_port_data_eop	1	Output	It indicates the End-Of-Packet for this output port from the switch.
out_port_data_status	1	Output	It indicates the status of Packet for this output port from the switch. NO_ERR; FCS_ERR

D. Packet Structure

A data packet on Ethernet is called a Ethernet packet structure, which transmits a Ethernet frame as payload. An Ethernet frame is preceded by a preamble and start frame delimiter (SFD), which are both part of the layer 1 Ethernet packet. Each Ethernet frame starts with an Ethernet header, which contains destination and source addresses as its first two fields. The middle section of the frame is payload data including any headers for other protocols carried in the frame. The frame ends with a frame check sequence (FCS), which is a 32-bit cyclic redundancy check used to detect any in-transit corruption of data. A data packet on the wire and the frame as its payload consist of binary data.

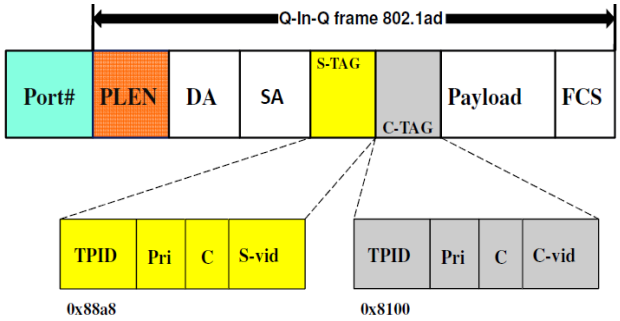


Fig. 4 Packet Structure

Output port number is of 6 bits width. Packet length (PLEN) is 10 bits width including header and payload. Destination address (DA) and Source address(SA) is of 6 bytes width. Service tag (STAG) and Customer tag (CTAG) and Frame check sequence (FCS) is 4 bytes of width. PLEN specifies the size of payload. Tag protocol identifier (TPID) determines whether a data frame contains a VLAN tag, the length of TPID is 16 bits. The default value is 0x88A8 for STAG and 0x8100 for CTAG. Pri represents the 802.1P priority of a frame, which determine how the frame should be queued and scheduled .It gives information about, whether the frame can be discarded or not in the event of traffic congestion The length of the Priority is 3 bits. C identifies whether the MAC address is encapsulated in the standard format in different transmission mediums. The length of the CFI is 1 bit. The value 0 indicates the MAC address is encapsulated in the standard format. The value 1 indicates that the MAC address is not encapsulated in the standard format. The default value is 0. S-vid and C-vid Identifies the number of the VLAN to which a frame belongs. The length of the VLAN ID is 12 bits. The value ranges from 0 to 4095. Because 0 and 4095 are reserved values of the protocol, the VLAN ID ranges from 1 to 4094.

V. GRAPH MODEL

The best way to represent functionality is with a set of graph-based scenario models. The graph captures the data flow paths and documents how to configure the blocks to perform all the operations that is designed to do. Constraints on the graph guide the generator and keep it from producing test cases that don't reflect intended behavior. The graph-based scenario models provide all the information needed to generate an unlimited number of multi-threaded test cases. These test cases are generated from graph-based scenario models that capture intended system behaviour.

©IJRASET 2015: All Rights are Reserved

Volume 3 Issue IX, September 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

A. Scenario Model

Scenario models simplify the generation of test cases. They consist of goals and their relationship with each other. Tests get generated by walking a scenario model. The arrows in the scenario model point to the goal's children.

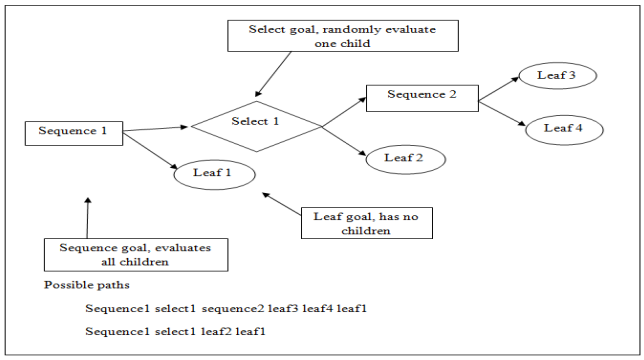


Fig. 5 Scenario Model

There are three types of goals sequence, select and leaf goal. A sequence goal evaluates a children in sequence.

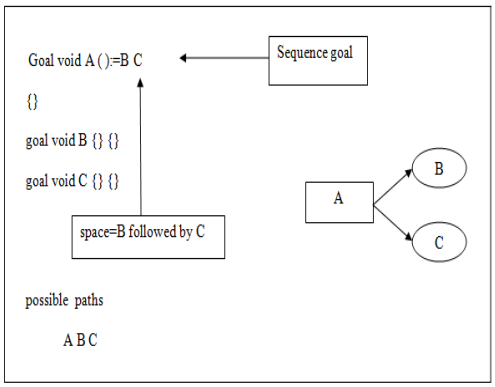


Fig. 6 Sequence Goal

A select goal randomly evaluates one of its children, leaving the remaining unevaluated. A leaf goal has no children.

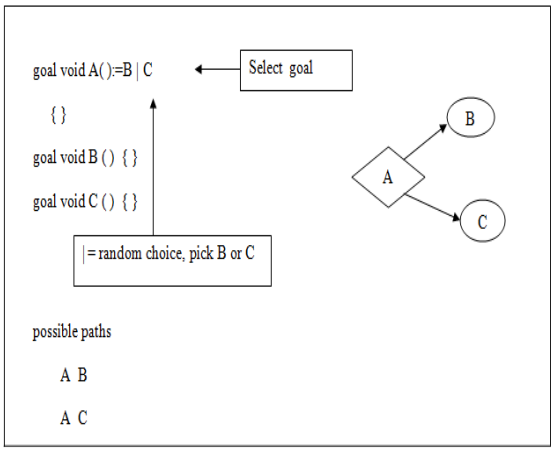


Fig. 7 Select Goal

©IJRASET 2015: All Rights are Reserved

Volume 3 Issue IX, September 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

B. Goals

A scenario model is a control structure made up of nodes called goals. The parent-child relationships between these Goals define the structure of the scenario model. A goal may have arguments and a return type. It also can have an initialization block and a body. The initial block is enclosed with square brackets, [ ], while the body is enclosed with curly braces, { }. A goal's children, initial block, and body may be redefined. If we want to change when children are evaluated, we can use the trek_evaluate_children() in select goals, for sequence goals it is better to add additional children in the sequence. There is an alternative syntax for the sequence and select goals by using the keywords sequence and select, followed by child declarations that back-link the child (B and C) to the parent (A).

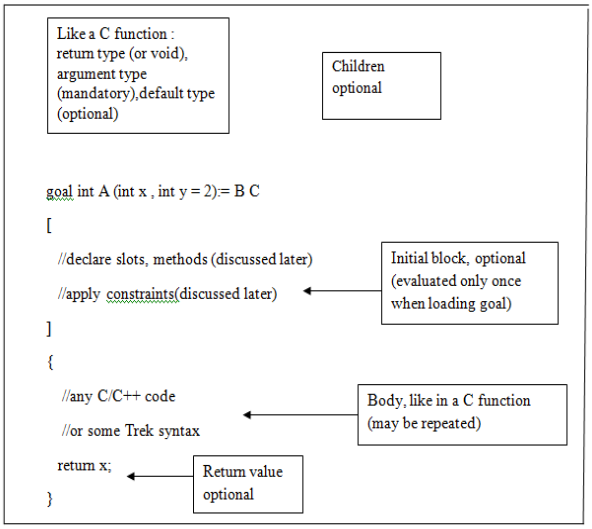


Fig. 8 Goal Declaration

C. Repeating And Weighing Goal

For sequence goals, the number of times that its child goal gets repeated can be specified. For any children of a sequence goal you place a colon (:) followed by a repeat number.

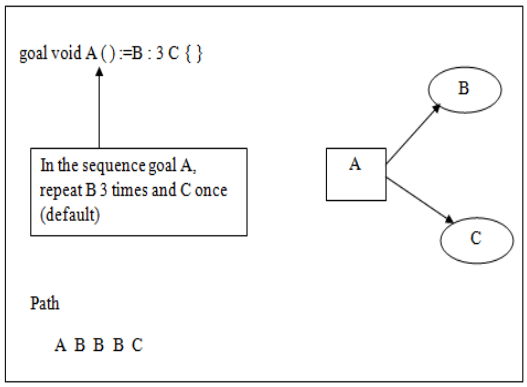


Fig. 9 Repeating Goal

For a select goal a weigh of any of its children can specify by placing a colon (:) followed by a weight number.

©IJRASET 2015: All Rights are Reserved

Volume 3 Issue IX, September 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

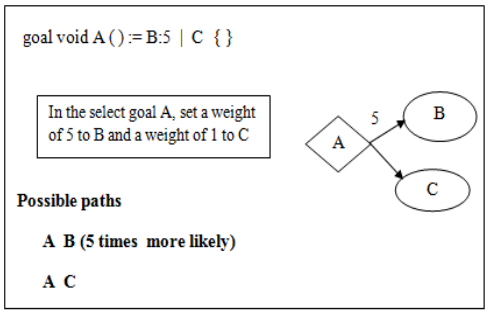
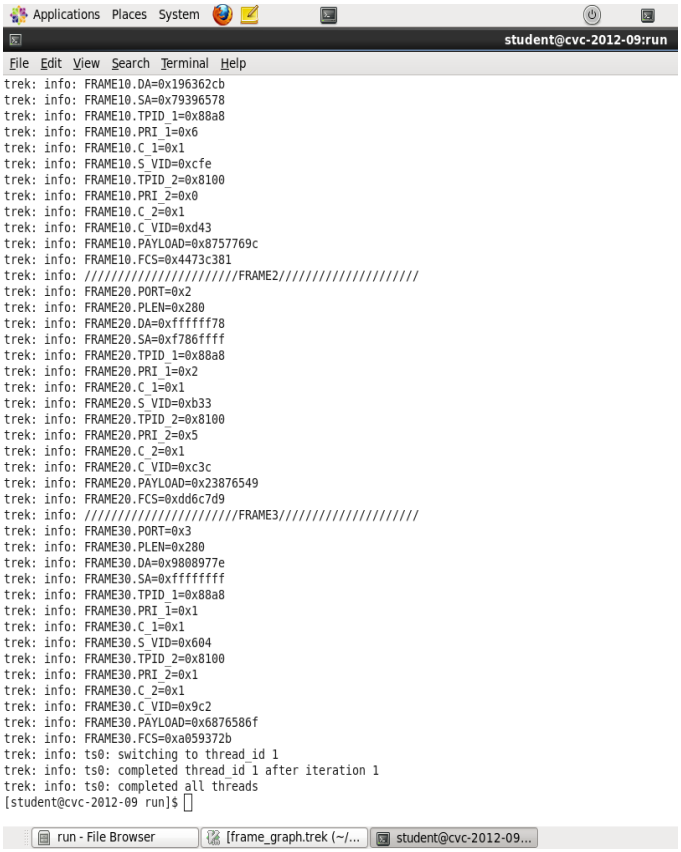


Fig. 10 Weighing Goal

VI. CONCLUSIONS

This paper introduces how to generate a graph in Trek software for Ethernet parser and router. The basic functionality and operation of Ethernet Parser and Router, description of signals and pins were discussed. Functional verification of Ethernet contains the description of verification platform using Verilog for the design under test. Testcases are generated automatically from the graph. In future the Graph is generated to transmit and receive date for Ethernet parser and router. Testcases generated automatically from the graph is given as input and Verilog coding is used to integrate with the testbench. It is then integrated with Device Under Test (DUT) and hence verification is done.



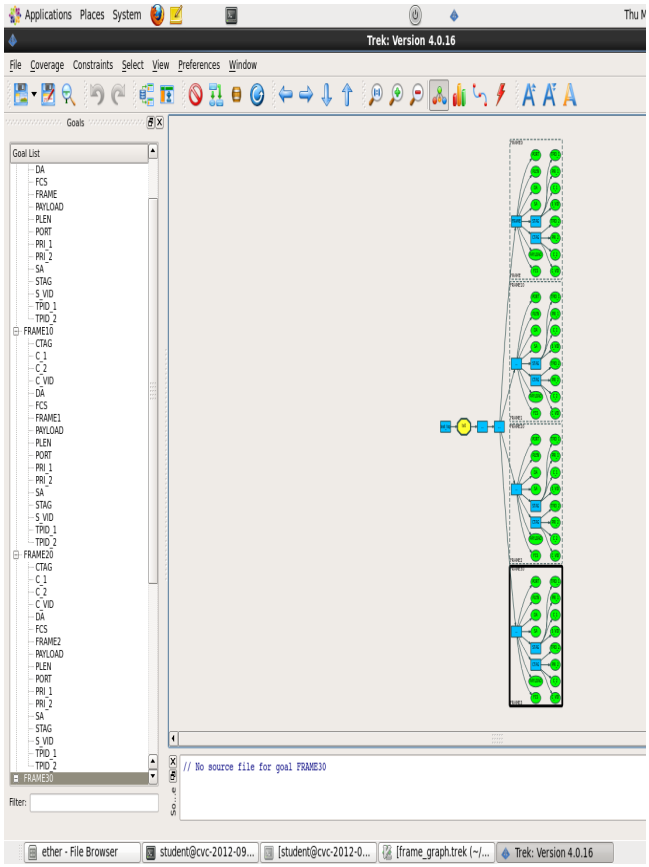


Fig. 11 Output value for Frame Structure

Fig. 12 Graph for Frame Structure

VII. ACKNOWLEDGMENT

©IJRASET 2015: All Rights are Reserved

I would like to express my sincere thanks to Michael Shell and other contributors for developing and maintaining the IEEE LaTeX style files which have been used in the preparation of this template.

REFERENCES

[1] Andreas Meyer, Verification Technologist, Mentor Graphics Corporation, “Graph-Based IP Verification in an ARM SoC Environment”.
[2] Dennis Ramaekers, “Graph-IC Verification”.
[3] Mohammad Reza Kakoee , M.H. Neishaburi , Siamak Mohammadi, “Graph based test case generation for TLM functional verification” , Received 15 November 2007, Revised 5 February 2008, Accepted 5 March 2008.
[4] Rion Hollenbeck ICS 620 Dr. Jones, “The IEEE 802.3 Standard (Ethernet): An Overview of the Technology”, 17 September, 2001.

©IJRASET 2015: All Rights are Reserved