FPGA BASED HARDWARE FIREWALL

By

Zaheer Ahmed Gauhar, Muaaz Bin Zahid, Jawad Riaz, M. Talha Tariq

Submitted to the Faculty of Department of Electrical Engineering,

Military College of Signals, National University of Science and Technology, Islamabad

in partial fulfillment for the requirement of a B.E Degree in

Telecom Engineering

JUNE 2016

CERTIFICATE

This is to certify that this project report entitled "**FPGA based Hardware Firewall"** by **Zaheer Ahmed Gauhar, Muaz Bin Zahid, Jawad Riaz, M. Talha Tariq**, submitted in partial fulfillment of the requirement for the degree of BE (Electrical Telecom) in Military College of Signals (NUST) during the academic year 2015-16, is a bona fide record of work carried out under my guidance and supervision.

Name (Supervisor):	**Asst. Prof. Mian Muhammad Waseem Iqbal**
Signature:	
Date:	

ABSTRACT

The basic aim of the project is to design and implement a Hardware Firewall which would enable user to secure a communication Network. The hardware firewall will use packet filtering technique to examine the header of a packet to determine its source and destination. This information will be compared to a set of predefined or user-created rules that will determine whether the packet is to be forwarded or dropped. Packet-filtering firewalls provide a reasonable amount of protection for a network with minimum complications. Packet-filtering rules can be extremely intuitive and thus easy to set up.

Copyright Notice:

Copyright in text of this thesis rests with the student author. Copies (by any process) either in full, or of extracts, may be made **only** in accordance with instructions given by the author and lodged in the Library of Military College of Signals(MCS), NUST. Details may be obtained by the Librarian. This page must form part of any such copies made. Further copies (by any process) of copies made in accordance with such instructions may not be made without the permission (in writing) of the author.

The ownership of any intellectual property rights which may be described in this thesis is vested in Military College of Signals(MCS), NUST, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the Military College of Signals(MCS), NUST, which will prescribe the terms and conditions of any such agreement. Further information on the conditions under which disclosures and exploitation may take place is available from the Library of Military College of Signals (MCS), NUST, Rawalpindi.

DEDICATION

This dissertation is gratefully dedicated to our parents and teachers. We could not have completed our study without their love, encouragement, prayers and guidance.

ACKNOWLEDGEMENTS

This project would not have been accomplished without Allah Almighty's will. We humbly thank Him for His blessings and giving us the wisdom, knowledge and understanding, without which we would not have been able to complete this thesis research work.

Due extension of gratitude to our project supervisor, Asst. Prof. Mian M. Waseem Iqbal, without his support and encouragement; it would not have been possible to complete this project.

We also thank our colleagues for helping us in developing the project and appreciate the people who have willingly helped us with their abilities.

Last but not the least; we are very thankful to our parents, who bore with us in times of difficulty and helped us overcoming obstacles. Without their consistent support and encouragement, we could not have accomplished our targets successfully.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES:

KEY TO SYMBOL OR ABBREVIATION:

➢ **FPGA**	Field Programmable Gate Array
➢ **MAC**	Media Access Control
➢ **IP**	Internet Protocol
➢ **TEMAC**	Tri Ethernet Media Access Control
➢ **MII**	Media Independent Interface
➢ **GMII**	Giga bit Media Independent Interface

CHAPTER 1:

INTRODUCTION:

1.1 Overview:

A Hardware *firewall* is basically a protective barrier. A standalone hardware firewall implementation on FPGA kit ensures secure communication. It will be inserted in between any network system and the internet and a controlled link will be established with a single checkpoint where security will be imposed. Security is ensured by checking IP packets entering the user network. The hardware firewall will be packet filtering to examine the header of a packet to determine its source and destination. This information will be compared to a set of predefined or user-created rules that will determine whether the packet is to be forwarded or dropped.

1.2 Problem Statement:

Internet connectivity is a necessity in modern age. Increased use of internet calls for extreme security measures. One of them is firewall. Firewall can either be hardware or software.

Software solutions are not that effective due to following reasons: o Slow due to sequential execution of algorithms.

- o They can be hacked easily by an attack on the computer it is running within. To make them fool-proof, a lot of time and investment is needed.
- o Cannot be customized according to the customer's and/or organization's needs.

On the contrary, hardware firewall has many advantages which include:

1. Fast processing and filtering due to dedicated resources (Parallel Processing)
2. Customizability
3. Single central solution
4. Easier to maintain and ideal for large organizations

1.3 Approach:

Any network system can be secured effectively using hardware means and the implementation of hardware firewall on a FPGA (Field Programmable Gate Array) board using Verilog language is one of them. By using FPGA boards, we can reconfigure digital integrated circuit for a hardware system to filter the IP packets and thus ensuring a secure incoming of data inside any network. FPGA boards are used as they have real-time applications with massively high parallel data processing. With FPGA, systems can be developed that are exactly customized for the designated task. It is relatively resistant to any cyber-attack as the whole device is hardware based. We aim to produce the hardware solution for any specific number of users belonging to a specific network.

The following figure sums up the whole process for filtering the data:

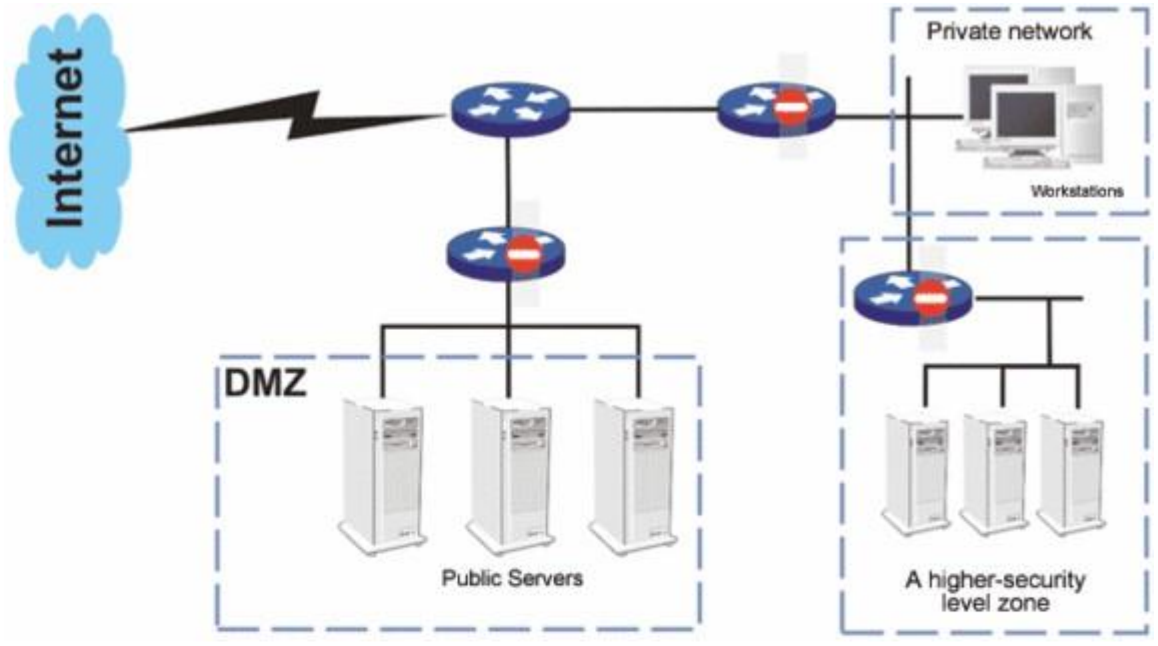


Figure 1. Hardware Firewall Implementation

Users of the network will be connected through Ethernet port on FPGA board. All the ports will be configured through Xilinx Software. The network firewall will compare the incoming IP packets (Ethernet and IP headers) to those in the list. If the IP packet is in the allowable list, the packet is processed ahead, otherwise discarded.

The Ethernet ports of the FPGA board will be designed and configured using Xilinx Software. In FPGA board we will define policies for every user based on its IP fields.

1.4 Objective:

The basic aim of the project is to

1. Design and implement a Hardware Firewall which would enable to secure a communication Network.
2. The project will be able to expose "hostile" elements and will be able to withstand these Intruder attacks.
3. The hardware firewall will be packet filtering to examine the header of a packet to determine its source and destination IP address and ports.
4. The hardware built will be strong enough to support a large number of users over a network.
5. Hardware firewall would be performing the following functions:
    - Packet Filtering
    - Centralized firewall solution

The main block will be built using Verilog Hardware Description Language (HDL) to speed up the packet processing. A processor based embedded system with real-time operating system will be designed to achieve highly customized and on-the-fly configuration change in the firewall.

CHAPTER 2:

BACKGROUND SCOPE:

2.1 Existing Literature:

A computer firewall protects networked computers from intentional hostile intrusion that could compromise confidentiality or result in data corruption or denial of service. It may be a hardware device or a software program running on a secure host computer. In either case, it must have at least two network interfaces, one for the network it is intended to protect, and one for the network it is exposed to. A network firewall sits at the junction point or gateway between the two networks, usually a private network and a public network such as the Internet. The earliest computer firewalls were simple routers.

There are two access denial methodologies used by computer firewalls. A firewall may allow all traffic through unless it meets certain criteria, or it may deny all traffic unless it meets certain criteria. The type of criteria used to determine whether traffic should be allowed through varies from one type of firewall to another. Computer Firewalls may be concerned with the type of traffic, or with source or destination addresses and ports.

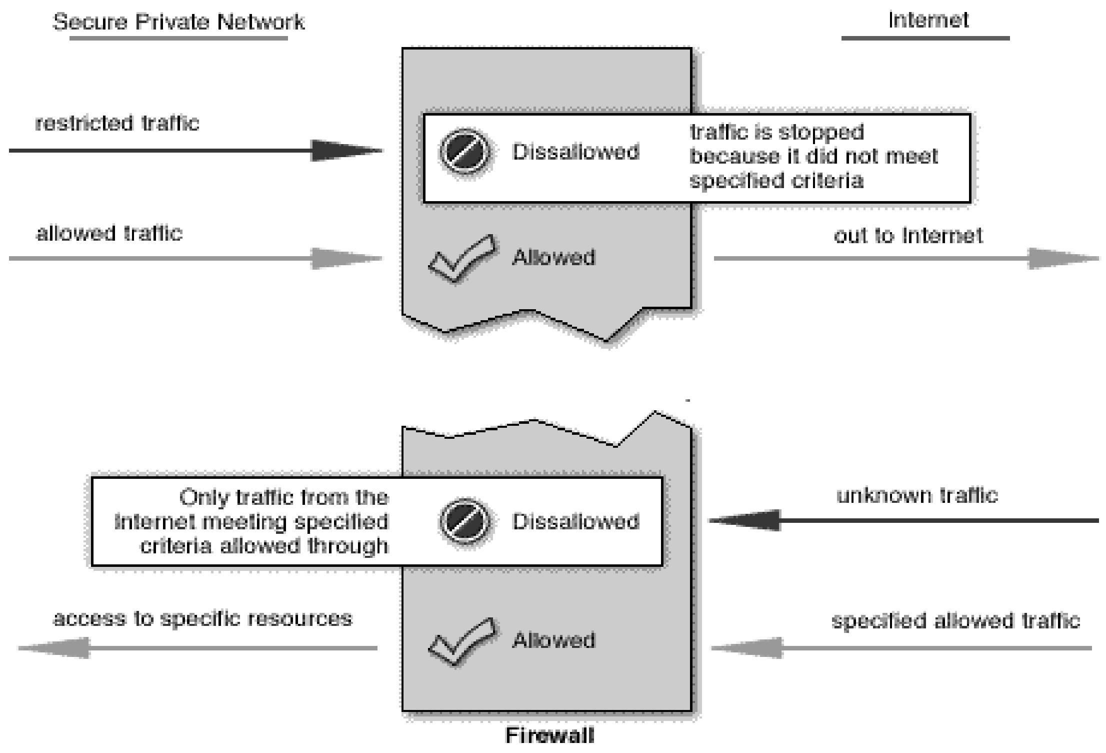

Secure Private Network
Internet
restricted traffic
Dissallowed
traffic is stopped because it did not meet specified criteria
allowed traffic
Allowed
out to Internet
Only traffic from the Internet meeting specified criteria allowed through
Dissallowed
unknown traffic
access to specific resources
Allowed
specified allowed traffic
Firewall

Figure 2. Basic Firewall Functioning

2.1.1 Types of Firewall:

Firewalls fall into four broad categories: Packet Filters, Circuit Level Gateways, Application Level Gateways and Stateful Multilayer Inspection Firewalls.

Packet filtering firewalls work at the network layer of the OSI model, or the IP layer of TCP/IP. They are usually part of a router. A router is a device that receives packets from one network and forwards them to another network. In a packet filtering firewall each packet is compared to a set of criteria before it is forwarded. Depending on the packet and the criteria, the firewall can drop the packet, forward it or send a message to the originator. Rules can include source and destination IP address, source and destination port number and protocol used. The advantage of packet filtering firewalls is their low cost and low impact on network performance. Most routers support packet filtering. Even if other firewalls are used, implementing packet filtering at the router level affords an initial degree of security at a low network layer. This type of firewall only works at the network layer, however, and does not support sophisticated rule based models. Network

Address Translation (NAT) routers offer the advantages of packet filtering firewalls but can also hide the IP addresses of computers behind the firewall, and offer a level of circuit-based filtering.

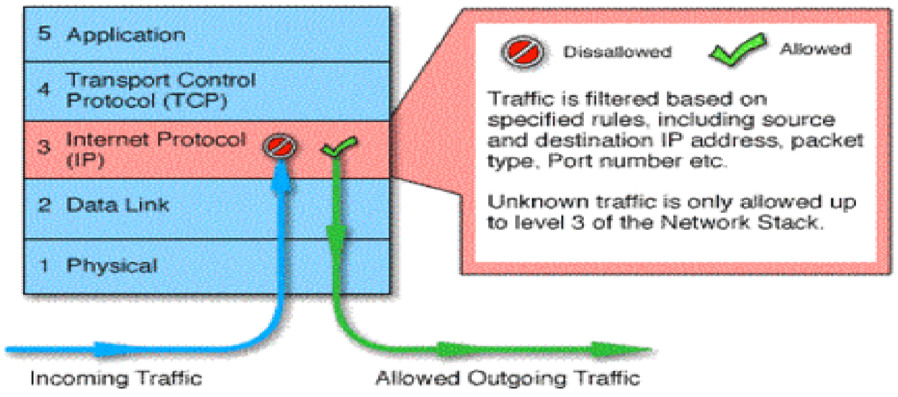

5 Application
4 Transport Control Protocol (TCP)
3 Internet Protocol (IP)
2 Data Link
1 Physical
Dissallowed
Allowed
Traffic is filtered based on specified rules, including source and destination IP address, packet type, Port number etc.
Unknown traffic is only allowed up to level 3 of the Network Stack.
Incoming Traffic
Allowed Outgoing Traffic

Figure 3. Packet Filtering Firewall

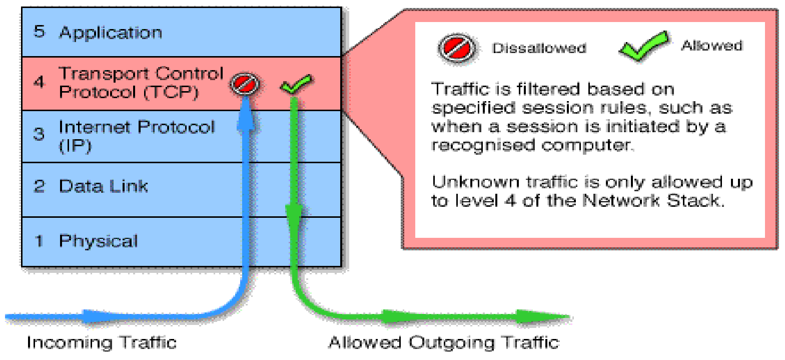

5 Application
4 Transport Control Protocol (TCP)
3 Internet Protocol (IP)
2 Data Link
1 Physical
Dissallowed
Allowed
Traffic is filtered based on specified session rules, such as when a session is initiated by a recognised computer.
Unknown traffic is only allowed up to level 4 of the Network Stack.
Incoming Traffic
Allowed Outgoing Traffic

Figure 4. Circuit Level Gateway

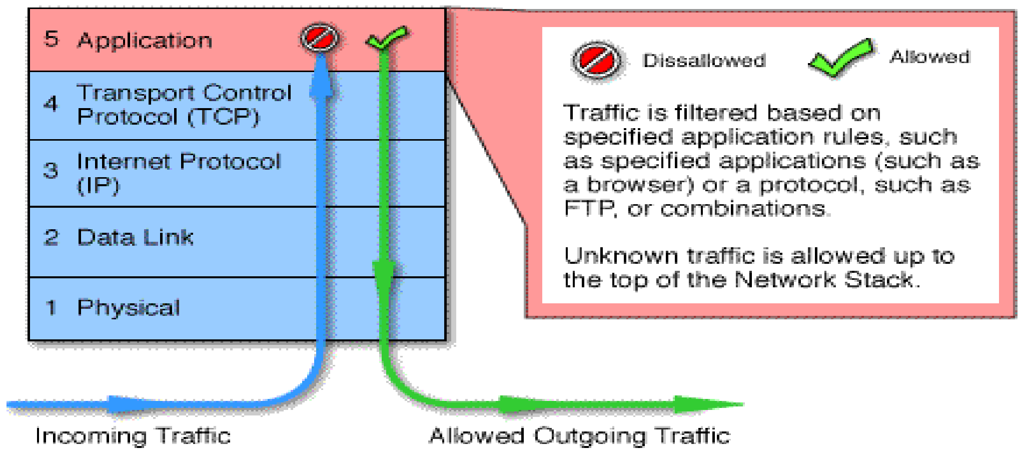

5 Application
4 Transport Control Protocol (TCP)
3 Internet Protocol (IP)
2 Data Link
1 Physical
Incoming Traffic
Allowed Outgoing Traffic
Dissallowed
Allowed
Traffic is filtered based on specified application rules, such as specified applications (such as a browser) or a protocol, such as FTP, or combinations.
Unknown traffic is allowed up to the top of the Network Stack.

Figure 5. Application Level Gateway

2.2 Problem Formulation:

Some work has already been done for the hardware firewall's FPGA implementation. Almost all of them make use of FPGAs' flexibility and architecture to design systems with increased speed and optimized silicon area usage. We will be using Xilinx Virtex-6 Development to design the modules as it has 100MHz fixed oscillator and up to 400MHz programmable clock generator. Furthermore, it has GTP transceiver, which can provide complete 1000 Base-X implementation on-chip. It is out aim to make a device that fully exploits the flexibility and architecture of FPGA to provide a faster and efficient solution to secure any network from any malware.

CHAPTER 3:

DESIGN:

3.1 Technical Specifications:

We would be using Xilinx Virtex-5 XC5VLX110T Development kit for our project.

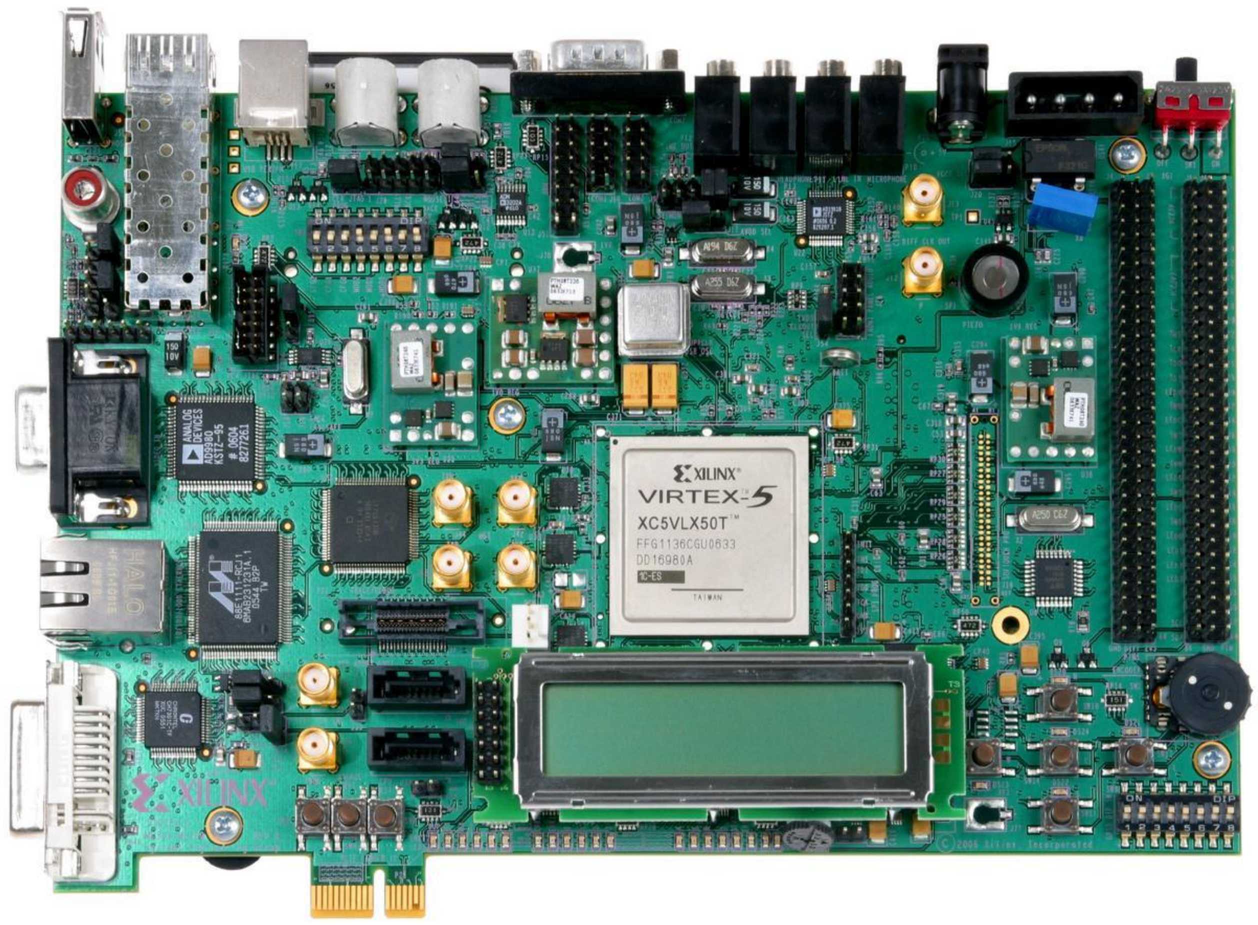


Figure 6. Virtex 5 FPGA Development Board

The hardware specifications of the board are:

- Device operation at less than 30W – 5V DC
- voltage-controlled oscillator (VCO) with a frequency from 600 MHz up to 1600 MHz

- 10/100/1000 Ethernet PHY.
- Each Virtex-6 FPGA slice contains four LUTs and eight flip-flops, only some slices can use their LUTs as distributed RAM or SRLs.
- GPIO includes 8 LEDs, 2 buttons, 2-axis navigation switch, 8 slide switches and a 16x2 LCD

Virtex-6 devices contain up to eight embedded Ethernet MACs, two per Ethernet MAC block. The blocks have the following characteristics:

- Designed to the IEEE 802.3-2002 specification
- RGMII/GMII Interface with SelectIO or SGMII interface when used with RocketIO transceivers
- Half or full duplex
- Supports Jumbo frames

1000 Base-X PCS/PMA: When used with RocketIO

3.2 Verilog Modules:

In Verilog, circuit components are designed inside a module. Modules can contain both structural and behavioral statements. Structural statements represent circuit components like logic gates, counters, and microprocessors. Behavioral level statements are programming statements that have no direct mapping to circuit components like loops, if-then statements, and stimulus vectors which are used to exercise a circuit. The module is the basic unit of hierarchy in Verilog. Modules describe:

- Boundaries (module, endmodule)
- Inputs and outputs (ports)
- Working behavior (behavioral or RTL code)
- Can be a single element or collection of lower level modules Module can describe a hierarchical design (a module of modules)

3.3 Control Module:

The type of coding which we are using in Verilog Programming is Behavioral coding with device XC6VLX240T having package FF1156, because behavioral Verilog code generally looks more like a sequential computer program running from top to bottom with an initial begin block. There are no input/output ports defined in top level behavioral modules since the output be to simulator facilities like the console display() etc. While the other type of coding is RTL (Register Transfer Level), in which code is defined in terms of registers and combinational logics that sits between them and intended to be used as input to a synthesis tool. Moreover it is a low level language having no explicit delays.

- **SUB MODULE:**

Sub module is a module used inside another module. When a module is used inside another module, it is not called but instantiated.

- **TEST BENCH:**

A test bench supplies the signals and dumps the outputs to simulate a Verilog design or modules. It invokes the design under test, generates the simulation input vectors, and implements the system tasks to view/format the results of the simulation. It is never synthesized so it can use all Verilog commands.

- **TOP MODULE:**

Top module is the first block of the Verilog program that will be executed at once, when the program is executed. It will pass all the input data from Test Bench to the Write Module. It also contains the instantiations of Write and Read modules and all the FIFOs. It outputs the data read from FIFOs.

```
 module top (clk, rst, sof, rx_data, eof, dout );
```

- **WRITE MODULE:**

When we start the process of packet filtering through code on Xilinx, the input packets are first processed by Write Module which extracts destination MAC address, Source MAC address, Source IP address, Destination IP address, Source Port number and Destination Port number and writes them to their respective FIFOs. It maintains a counter to keep track of the start of each address. Start of frame (SOF) and End of Frame (EOF) are used to mark the start and end of the input frame respectively.

```
module SM_WR(clk, rst,sof,rx_data,
eof,we_dest_mac,we_src_mac,we_type, we_payload, we_eof) ;
```

- **READ MODULE:**

Once the filtering parameters have been written to the FIFOs, they are read by the Read Module. It then outputs the information for display on the simulation screen. The output of this module will be used to compare the filtering parameters to the pre stored rules of firewall in order to take decisions for the forwarding or dropping of packet.

```
module SM_Rd(
clk,rst,eof_out,empty,rd_dest_mac,rd_src_mac,rd_type,rd_payl
oad,rd_eof) ;
```

3.4 Design Requirements:

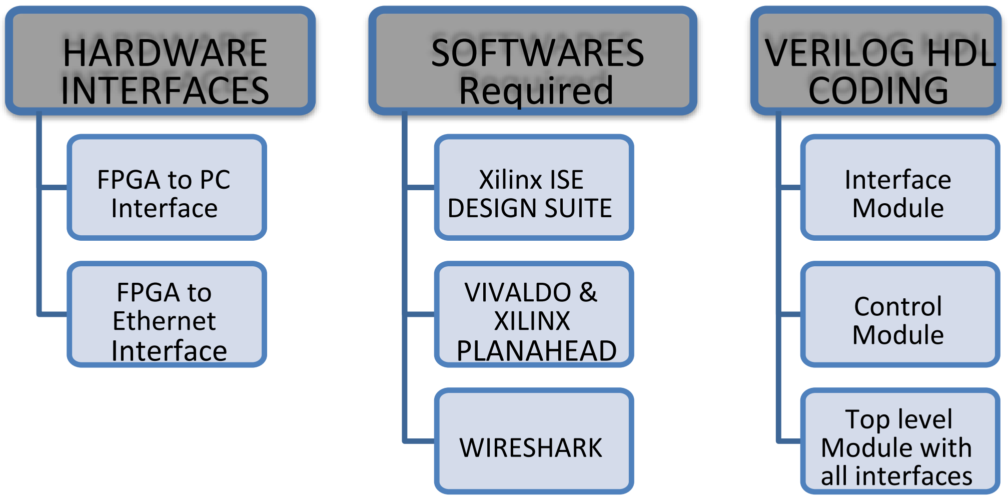


Figure 7. Requirements for the Design

3.5 Design Specifications:

We will design the Hardware Using Verilog HDL

- 1G MAC Control System will be incorporated in our system
- We will create a computer network and secure all the users within the network using single hardware firewall
- Throughput performance will be near line speed as we will be designing 1G MAC Control

- IP and Mac Address Filtering would be done based on user requirement and protocols being used
- Whole design will be in accordance to IEEE 802.3 Protocol

3.6 Detailed design with justification:

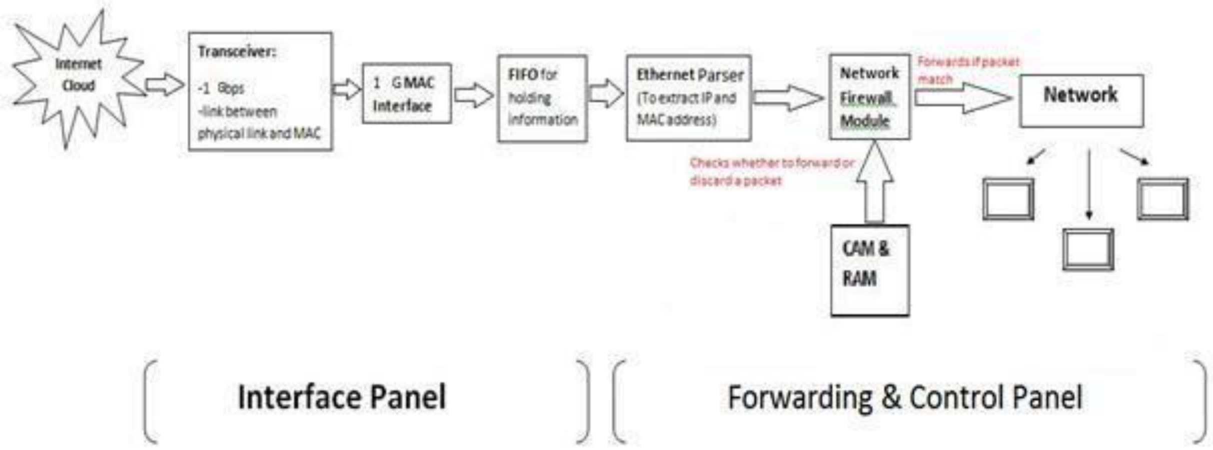

Internet Cloud
Transceiver:
-1 Gbps
-link between physical link and MAC
1 GMAC Interface
FIFO for holding information
Ethernet Parser (To extract IP and MAC address)
Network Firewall Module
Forwards if packet match
Network
Checks whether to forward or discard a packet
CAM & RAM
Interface Panel
Forwarding & Control Panel

Figure 8. Block Diagram of the Firewall

The whole design has been divided into 2 parts:

3.6.1 FPGA to Router Interface:

A wired Ethernet connection will be established through a CAT-5 cable between the RJ-45 connector of the FPGA and the RJ-45 connector of the router or the switch available.

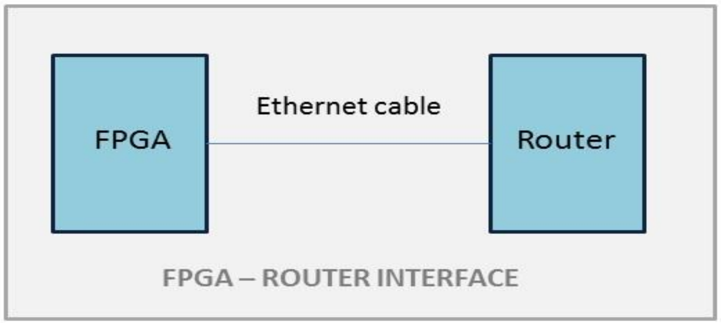


Figure 9. FPGA to Router Interface

3.6.2 1G Ethernet MAC Interface:

Enable Ethernet PHY of FPGA to transmit & receive data to and from the external Internet cloud. The steps involved are:

- Design of an Ethernet Core in Verilog HDL using the Xilinx Core generator and Xilinx Platform Studio.
- Modify the core using Xilinx Platform Studio according to the needs
- Import the core as a Xilinx project in Xilinx Project Navigator and generate .bit file
- Program the bit file on the FPGA using iMPAC software to enable the Ethernet MAC to send and receive the IP traffic in tri mode (10/100/1000Mbps)

1. Control Panel:

This panel basically decides whether a packet is to be forwarded to the network or discarded based on the IP and MAC addresses stored in the RAM by the network administrator. This panel thus controls the flow of information into the network. This panel will contain the following modules

a) IP filter

b) MAC Filter

2. Interface panel:

Responsible for the transfer and reception of signals to and from the physical interface. Interfacing Panel is being designed first after which Control panel will be designed. Below mentioned are the various Modules for the Interfacing Panel of Hardware Firewall.

3.7 Memory Module

In Xilinx FPGAs, a Block RAM is a dedicated two-port memory containing several kilobits of RAM. The FPGA contains several of these blocks. Inside of each small logic block is a configurable lookup table. It is normally used for logic functions, but you can reconfigure it as a few bits of RAM. You can combine several of them into a larger RAM. This is distributed RAM. Both types of RAM can be initialized with data, or used as ROM.

Distributed RAMs are implemented within Look up Tables (LUTs.) Each Configurable Logic Block (CLB) inside a Xilinx FPGA contains two slices and each slice contains 2 LUTs and LUTs are made of Static RAM (SRAM), which can accommodate 16 bits. So, for small memory distributed RAMs are used while for large memory Block RAM is used.

3.7.1 Block RAM Generator Core:

The Block Memory Generator core uses embedded Block Memory primitives in Xilinx FPGAs to extend the functionality and capability of a single primitive to memories of arbitrary widths and depths. Sophisticated algorithms within the Block Memory Generator core produce optimized solutions to provide convenient access to memories for a wide range of configurations.

3.7.2 Memory Initialization:

The memory contents are initialized using a memory coefficient (COE) file. A COE file defines the initial contents of each individual memory locations.

We have made a block RAM using Block RAM Core Generator to store set of rules for our Firewall, rules for destination MAC, source MAC, destination socket, source socket are stored in each individual Block RAM core.

3.7.3 Block Ram Instantiation:

The following instantiation shows the instantiation of block RAM core. In this core address is passed at addra and the data at that location is output to douta. The received data can then be compared to the contents of the received Ethernet frame, to decide whether to drop of forward the packet. This process is repeated until all the data values of Block RAM are being checked.

```
Dest_mac_core dest_mac(
 clka(clka),                    // clock of port A
 wea(wea),                      // write enable of port A
 addra(addra),                  // address of port A
 dina(dina),                    // data in at port A
 douta(dest_mac_dout)           // data out at port A
);
```

3.7.4 Top Module for Block RAM Simulation:

This module is used to pass the input signals from test bench to Block RAM cores. clka is used for synchronization purposes, rst is used for asynchronous reset, wea, addra, dina are the input signals for the Block RAM Core, dest_mac_data_out holds the output from destination MAC Block RAM core, src_mac_data_out holds the output from source MAC Block RAM core. Outputs from all the Block RAM cores are concatenated and checked against the corresponding parameters of Ethernet frame, if a match is found ‘match’ flag is set high indicating that the packet should be forwarded.

```
module Top(clka,                // clock of port A
rst,                            // reset
wea,                            // write enable port A
addra,                          // address of port A
dina,                           // data in at port A
dest_mac_data_out,              // reads destination MAC addresses from block
RAM
src_mac_data_out,               // reads source MAC addresses from block RAM
src_socket_data_out,            // reads source IP & port addresses from block RAM
```

dest_socket_data_out, // reads destination IP & port addresses from block RAM

match // 1 bit flag to respond when a match of rule is found

);

3.7.1 Memory Module Simulation:

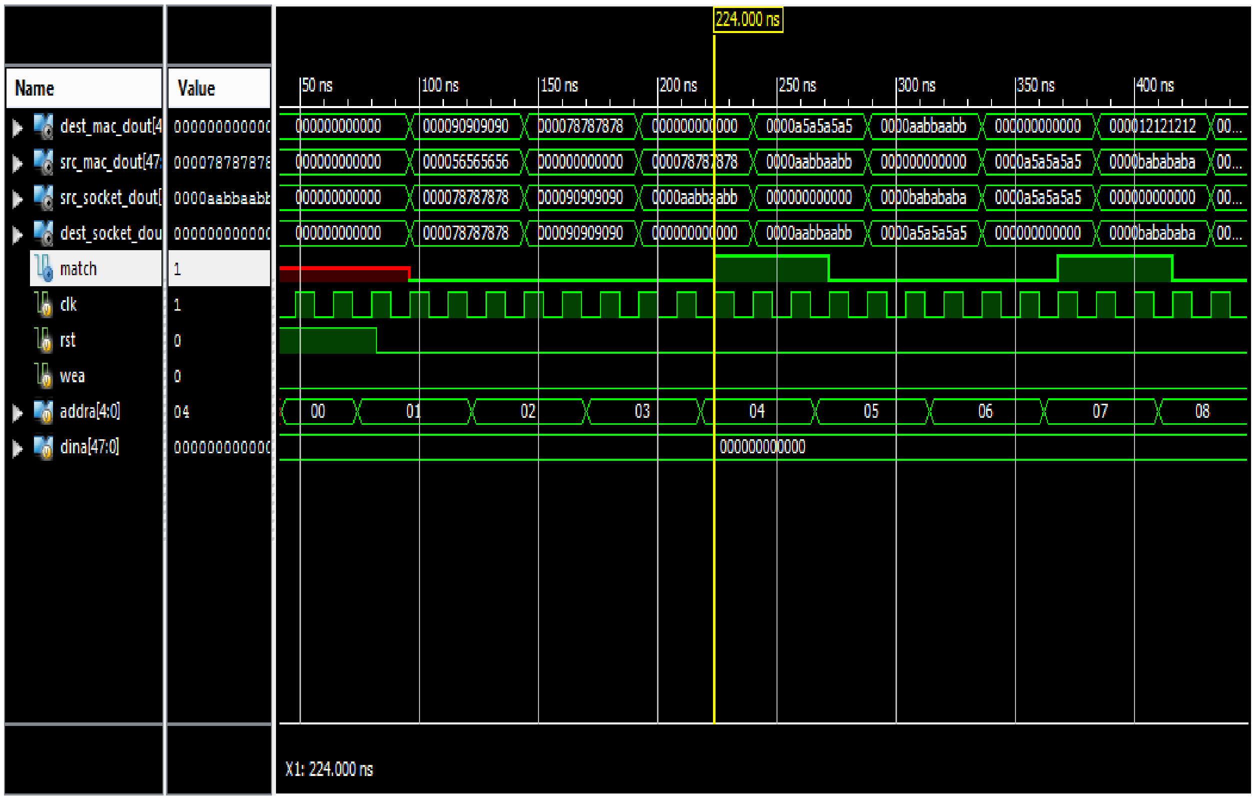

224.000 ns
Name
Value
dest_mac_dout[4
src_mac_dout[47
src_socket_dout[
dest_socket_dou
match
clk
rst
wea
addra[4:0]
dina[47:0]
000000000000
000078787878
0000aabbaabb
000000000000
1
1
0
0
04
000000000000
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
350 ns
400 ns
000000000000 000090909090 000078787878 000000000000 0000a5a5a5a5 0000aabbaabb 000000000000 000012121212 00...
000000000000 000056565656 000000000000 000078787878 0000aabbaabb 000000000000 0000a5a5a5a5 0000babababa 00...
000000000000 000078787878 000090909090 0000aabbaabb 000000000000 0000babababa 0000a5a5a5a5 000000000000 00...
000000000000 000078787878 000090909090 000000000000 0000aabbaabb 0000a5a5a5a5 000000000000 0000babababa 00...
00 01 02 03 04 05 06 07 08
000000000000
X1: 224.000 ns

Figure 10. Memory Module Simulation (Match Found)

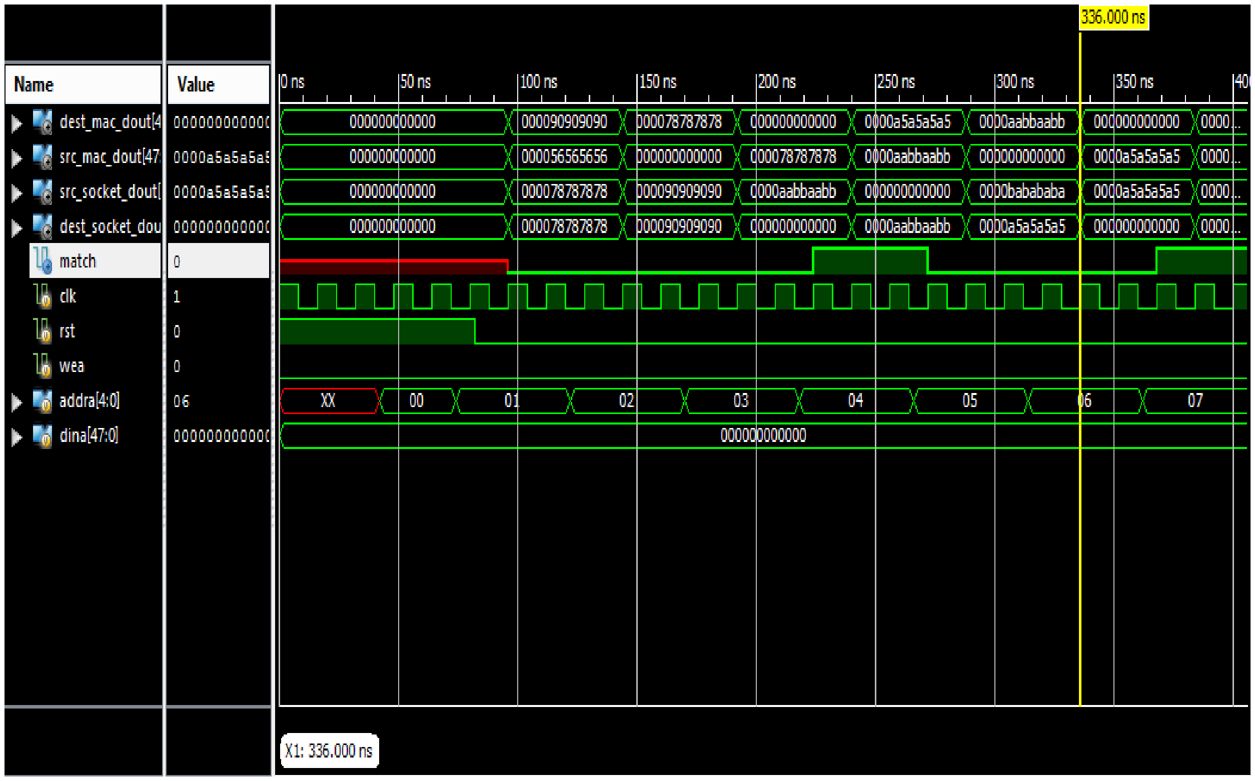


Figure 11. Memory Module Simulation (Match Not Found)

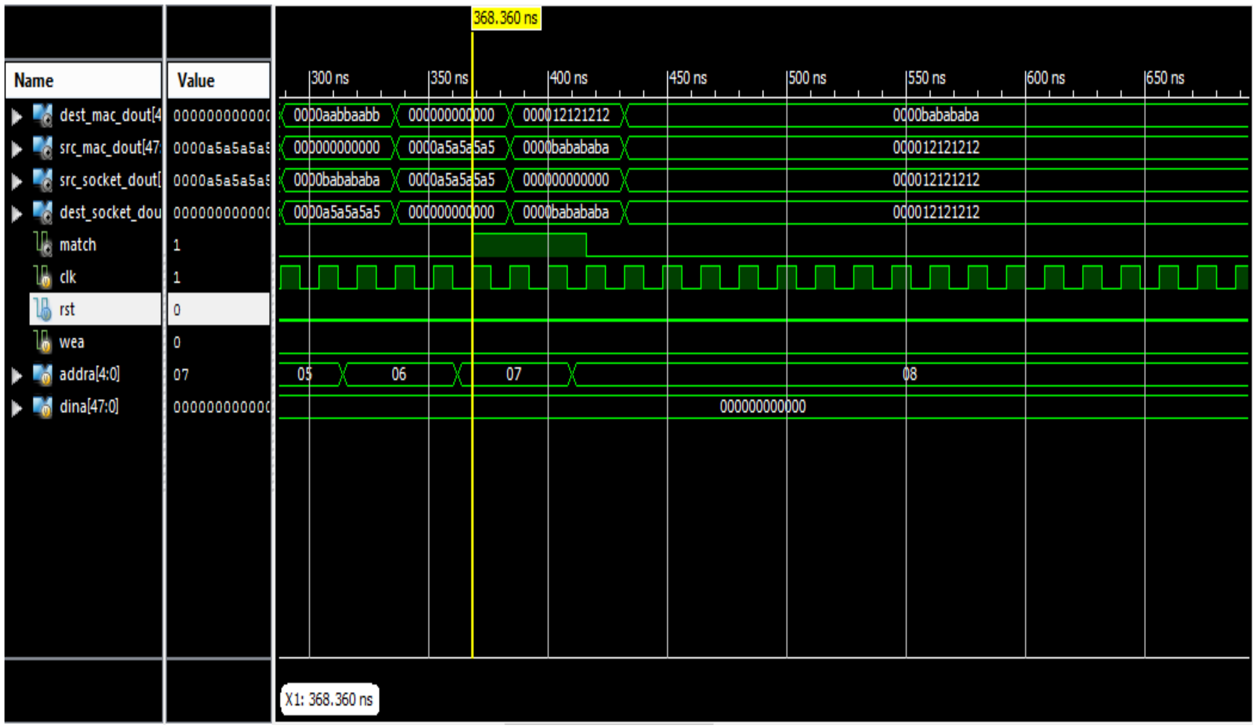


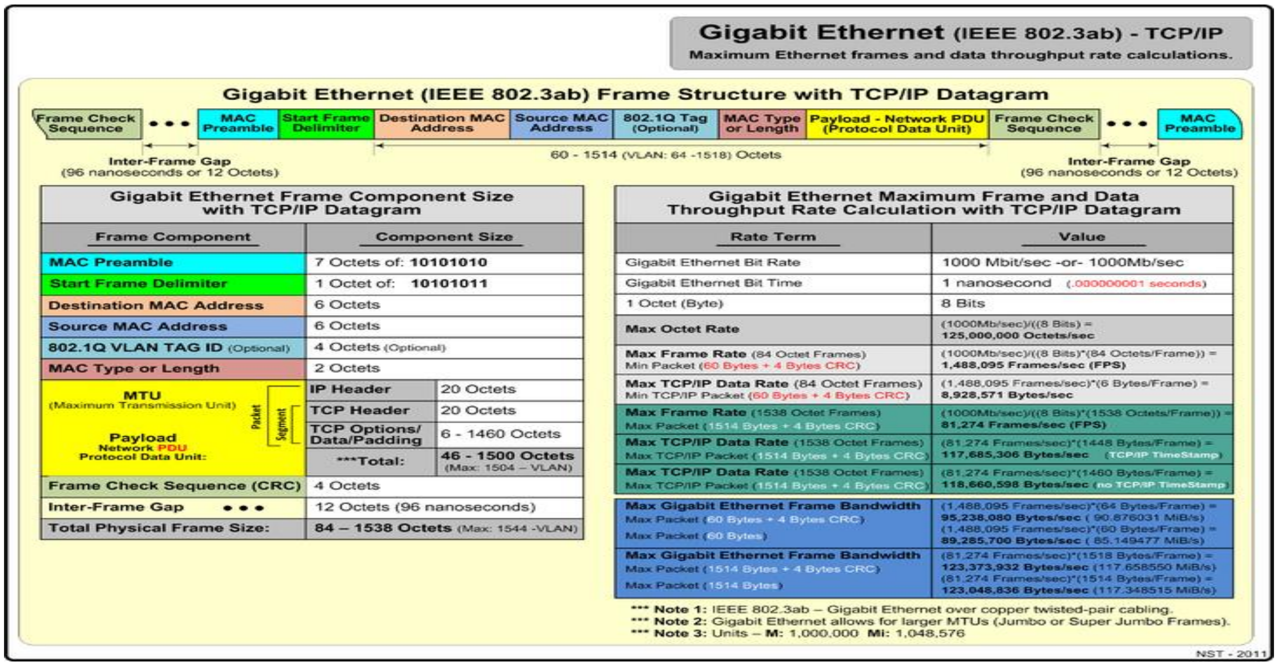
Figure 12. Memory Module Simulation (Match Found)

3.8 Resource Utilization:

Slice Logic Utilization	Used	Available	Utilization
No. of Slice Registers	952	69120	1%
No. used as Flip Flops	952	---	
No. of Slice LUT’s	884	69120	1%
No. used as logics	804		
No. used as Memory	70		
Total Memory used (KB)	144	5328	2%
No. of TEMACs	1	2	50%

Table 1: Resources Utilized

3.9 Frame Structure:


Gigabit Ethernet (IEEE 802.3ab) - TCP/IP
Maximum Ethernet frames and data throughput rate calculations.
Gigabit Ethernet (IEEE 802.3ab) Frame Structure with TCP/IP Datagram
Frame Check Sequence
MAC Preamble
Start Frame Delimiter
Destination MAC Address
Source MAC Address
802.1Q Tag (Optional)
MAC Type or Length
Payload - Network PDU (Protocol Data Unit)
Frame Check Sequence
MAC Preamble
Inter-Frame Gap (96 nanoseconds or 12 Octets)
60 - 1514 (VLAN: 64 -1518) Octets
Inter-Frame Gap (96 nanoseconds or 12 Octets)
NST - 2011

Gigabit Ethernet Frame Component Size with TCP/IP Datagram

Frame Component		Component Size	
MAC Preamble		7 Octets of: **10101010**	
Start Frame Delimiter		1 Octet of: **10101011**	
Destination MAC Address		6 Octets	
Source MAC Address		6 Octets	
802.1Q VLAN TAG ID (Optional)		4 Octets (Optional)	
MAC Type or Length		2 Octets	
MTU (Maximum Transmission Unit) Payload Network PDU Protocol Data Unit:	Packet	IP Header	20 Octets
	Segment	TCP Header	20 Octets
		TCP Options/ Data/Padding	6 - 1460 Octets
		***Total:	**46 - 1500 Octets** (Max: 1504 – VLAN)
Frame Check Sequence (CRC)		4 Octets	
Inter-Frame Gap • • •		12 Octets (96 nanoseconds)	
Total Physical Frame Size:		**84 – 1538 Octets** (Max: 1544 -VLAN)	

Gigabit Ethernet Maximum Frame and Data Throughput Rate Calculation with TCP/IP Datagram

Rate Term	Value
Gigabit Ethernet Bit Rate	1000 Mbit/sec -or- 1000Mb/sec
Gigabit Ethernet Bit Time	1 nanosecond (.000000001 seconds)
1 Octet (Byte)	8 Bits
Max Octet Rate	(1000Mb/sec)/((8 Bits) = **125,000,000 Octets/sec**
Max Frame Rate (84 Octet Frames) Min Packet (60 Bytes + 4 Bytes CRC)	(1000Mb/sec)/((8 Bits)*(84 Octets/Frame)) = **1,488,095 Frames/sec (FPS)**
Max TCP/IP Data Rate (84 Octet Frames) Min TCP/IP Packet (60 Bytes + 4 Bytes CRC)	(1,488,095 Frames/sec)*(6 Bytes/Frame) = **8,928,571 Bytes/sec**
Max Frame Rate (1538 Octet Frames) Max Packet (1514 Bytes + 4 Bytes CRC)	(1000Mb/sec)/((8 Bits)*(1538 Octets/Frame)) = **81,274 Frames/sec (FPS)**
Max TCP/IP Data Rate (1538 Octet Frames) Max TCP/IP Packet (1514 Bytes + 4 Bytes CRC)	(81,274 Frames/sec)*(1448 Bytes/Frame) = **117,685,306 Bytes/sec** (TCP/IP TimeStamp)
Max TCP/IP Data Rate (1538 Octet Frames) Max TCP/IP Packet (1514 Bytes + 4 Bytes CRC)	(81,274 Frames/sec)*(1460 Bytes/Frame) = **118,660,598 Bytes/sec** (no TCP/IP TimeStamp)
Max Gigabit Ethernet Frame Bandwidth Max Packet (60 Bytes + 4 Bytes CRC) Max Packet (60 Bytes)	(1,488,095 Frames/sec)*(64 Bytes/Frame) = **95,238,080 Bytes/sec** (90.876031 MiB/s) (1,488,095 Frames/sec)*(60 Bytes/Frame) = **89,285,700 Bytes/sec** (85.149477 MiB/s)
Max Gigabit Ethernet Frame Bandwidth Max Packet (1514 Bytes + 4 Bytes CRC) Max Packet (1514 Bytes)	(81,274 Frames/sec)*(1518 Bytes/Frame) = **123,373,932 Bytes/sec** (117.658550 MiB/s) (81,274 Frames/sec)*(1514 Bytes/Frame) = **123,048,836 Bytes/sec** (117.348515 MiB/s)

*** **Note 1:** IEEE 802.3ab – Gigabit Ethernet over copper twisted-pair cabling.
*** **Note 2:** Gigabit Ethernet allows for larger MTUs (Jumbo or Super Jumbo Frames).
*** **Note 3:** Units – **M:** 1,000,000 **Mi:** 1,048,576

Figure 13: Ethernet Frame Structure

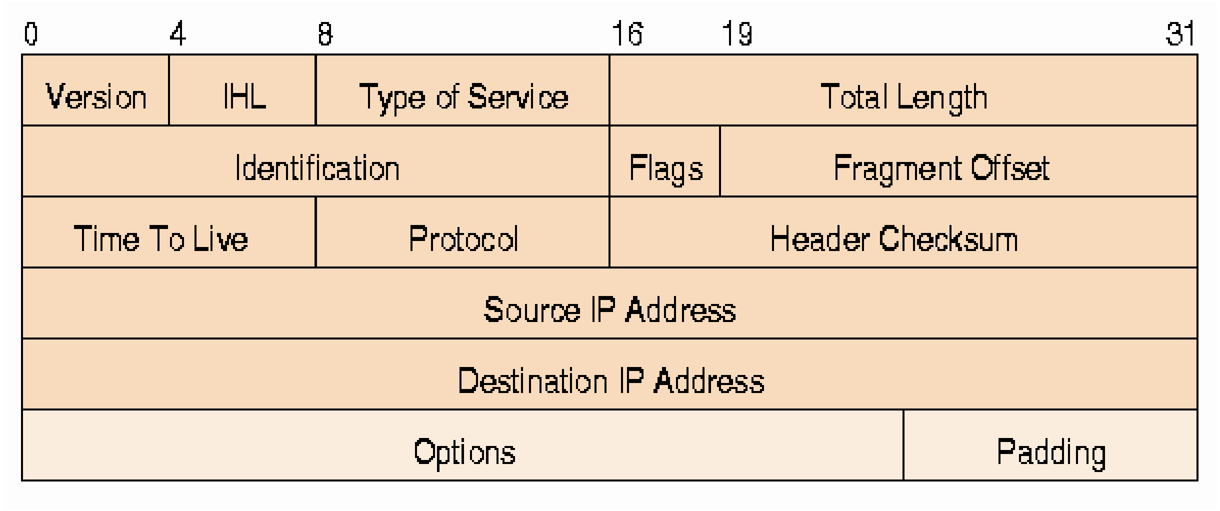
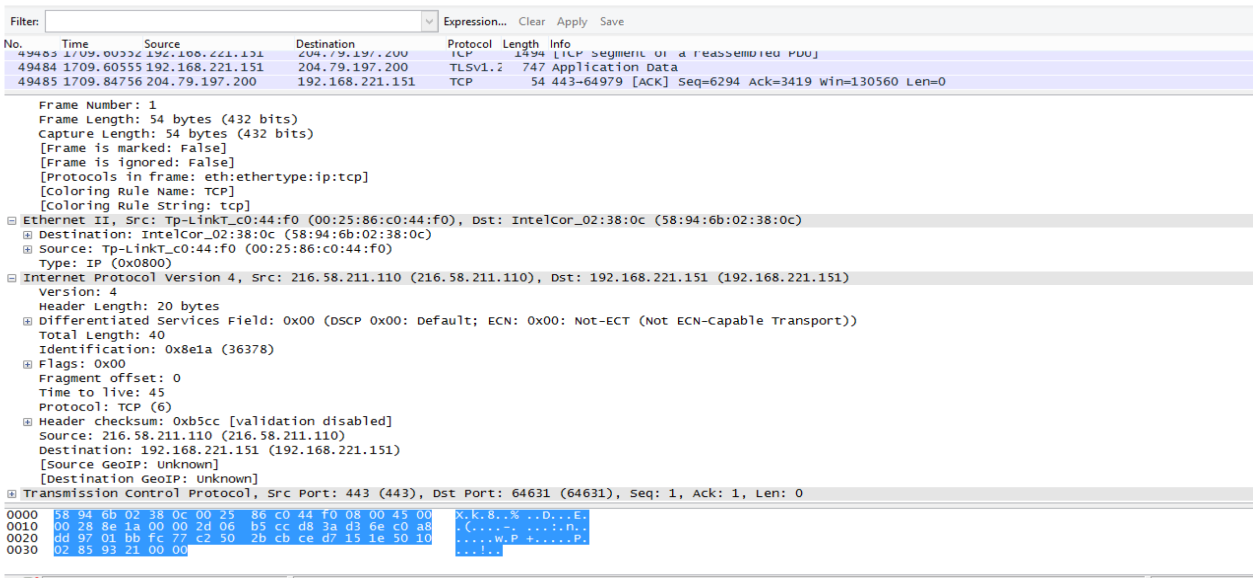


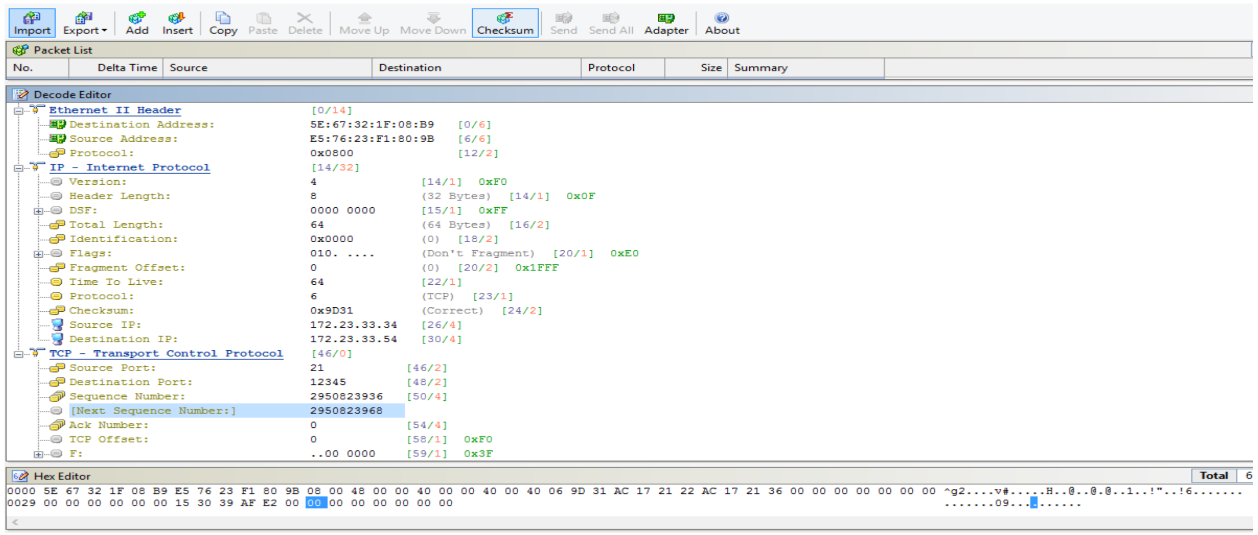
Figure 14: IP Packet Header

3.10 Packet in Wireshark and Packet Builder:



```
No.    Time       Source          Destination      Protocol Length Info
49483 1709.60552 192.168.221.151 204.79.197.200   TCP       1494 [TCP segment of a reassembled PDU]
49484 1709.60555 192.168.221.151 204.79.197.200   TLSv1.2    747 Application Data
49485 1709.84756 204.79.197.200  192.168.221.151  TCP         54 443→64979 [ACK] Seq=6294 Ack=3419 Win=130560 Len=0

    Frame Number: 1
    Frame Length: 54 bytes (432 bits)
    Capture Length: 54 bytes (432 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:tcp]
    [Coloring Rule Name: TCP]
    [Coloring Rule String: tcp]
⊟ Ethernet II, Src: Tp-LinkT_c0:44:f0 (00:25:86:c0:44:f0), Dst: IntelCor_02:38:0c (58:94:6b:02:38:0c)
  ⊞ Destination: IntelCor_02:38:0c (58:94:6b:02:38:0c)
  ⊞ Source: Tp-LinkT_c0:44:f0 (00:25:86:c0:44:f0)
    Type: IP (0x0800)
⊟ Internet Protocol Version 4, Src: 216.58.211.110 (216.58.211.110), Dst: 192.168.221.151 (192.168.221.151)
    Version: 4
    Header Length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 40
    Identification: 0x8e1a (36378)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 45
    Protocol: TCP (6)
  ⊞ Header checksum: 0xb5cc [validation disabled]
    Source: 216.58.211.110 (216.58.211.110)
    Destination: 192.168.221.151 (192.168.221.151)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
⊞ Transmission Control Protocol, Src Port: 443 (443), Dst Port: 64631 (64631), Seq: 1, Ack: 1, Len: 0

0000  58 94 6b 02 38 0c 00 25  86 c0 44 f0 08 00 45 00   X.k.8..% ..D...E.
0010  00 28 8e 1a 00 00 2d 06  b5 cc d8 3a d3 6e c0 a8   .(....-. ...:.n..
0020  dd 97 01 bb fc 77 c2 50  2b cb ce d7 15 1e 50 10   .....w.P +.....P.
0030  02 85 93 21 00 00                                  ...!..
```

Figure 15. Captured packet in Wireshark



```
No.   Delta Time  Source   Destination   Protocol   Size  Summary

⊟ Ethernet II Header                   [0/14]
    Destination Address:               5E:67:32:1F:08:B9   [0/6]
    Source Address:                    E5:76:23:F1:80:9B   [6/6]
    Protocol:                          0x0800              [12/2]
⊟ IP - Internet Protocol               [14/32]
    Version:                           4              [14/1]  0xF0
    Header Length:                     8              (32 Bytes)  [14/1]  0x0F
  ⊞ DSF:                               0000 0000      [15/1]  0xFF
    Total Length:                      64             (64 Bytes)  [16/2]
    Identification:                    0x0000         (0)  [18/2]
  ⊞ Flags:                             010. ....      (Don't Fragment)  [20/1]  0xE0
    Fragment Offset:                   0              (0)  [20/2]  0x1FFF
    Time To Live:                      64             [22/1]
    Protocol:                          6              (TCP)  [23/1]
    Checksum:                          0x9D31         (Correct)  [24/2]
    Source IP:                         172.23.33.34   [26/4]
    Destination IP:                    172.23.33.54   [30/4]
⊟ TCP - Transport Control Protocol     [46/0]
    Source Port:                       21           [46/2]
    Destination Port:                  12345        [48/2]
    Sequence Number:                   2950823936   [50/4]
    [Next Sequence Number:]            2950823968
    Ack Number:                        0            [54/4]
    TCP Offset:                        0            [58/1]  0xF0
  ⊞ F:                                 ..00 0000    [59/1]  0x3F

0000 5E 67 32 1F 08 B9 E5 76 23 F1 80 9B 08 00 48 00 00 40 00 00 40 00 40 06 9D 31 AC 17 21 22 AC 17 21 36 00 00 00 00 00 00 00 ^g2....v#.....H..@..@.@..1..!"..!6.......
0029 00 00 00 00 00 00 15 30 39 AF E2 00 00 00 00 00 00 00 00                                                                  .......09.........
```

Figure 16. TCP packet in Packet B

CHAPTER 4:

SIMULATION:

4.1 Test Bench:

A test bench supplies the signals and dumps the outputs to simulate a Verilog design or modules. It invokes the design under test, generates the simulation input vectors, and implements the system tasks to view/format the results of the simulation. It is never synthesized so it can use all Verilog commands.

4.2 Firewall MODULE:

```
module firewall (
     rx_ll_clock,                    // Input CLK from TRIMAC Reciever
     rx_ll_reset,                    // Synchronous reset signal
     rx_ll_data_in,                  // Input data
     rx_ll_sof_in_n,                 // Input start of frame
     rx_ll_eof_in_n,                 // Input end of frame
    rx_ll_src_rdy_in_n,              // Input source ready
    rx_ll_data_out,                  // Modified output data
    rx_ll_sof_out_n,                // Output start of frame
    rx_ll_eof_out_n,                // Output end of frame
   rx_ll_src_rdy_out_n,           // Output source ready
   rx_ll_dst_rdy_in_n            // Input destination ready     );
```

Input data is sent to 40 byte shift register as well as a state machine. Shift register holds the input data while state machine compares it to the pre-stored MAC, IP and PORT addresses. If the input data frame matches the pre-stored addresses, a forward flag (fwd) is set high. After traversing through shift register, if the fwd flag is high, the data is assigned to the rx_ll_data_out port. If a mismatch is found with any of the addresses, the fwd flag is set low and the frame is dropped.

4.2.1 Firewall Design:

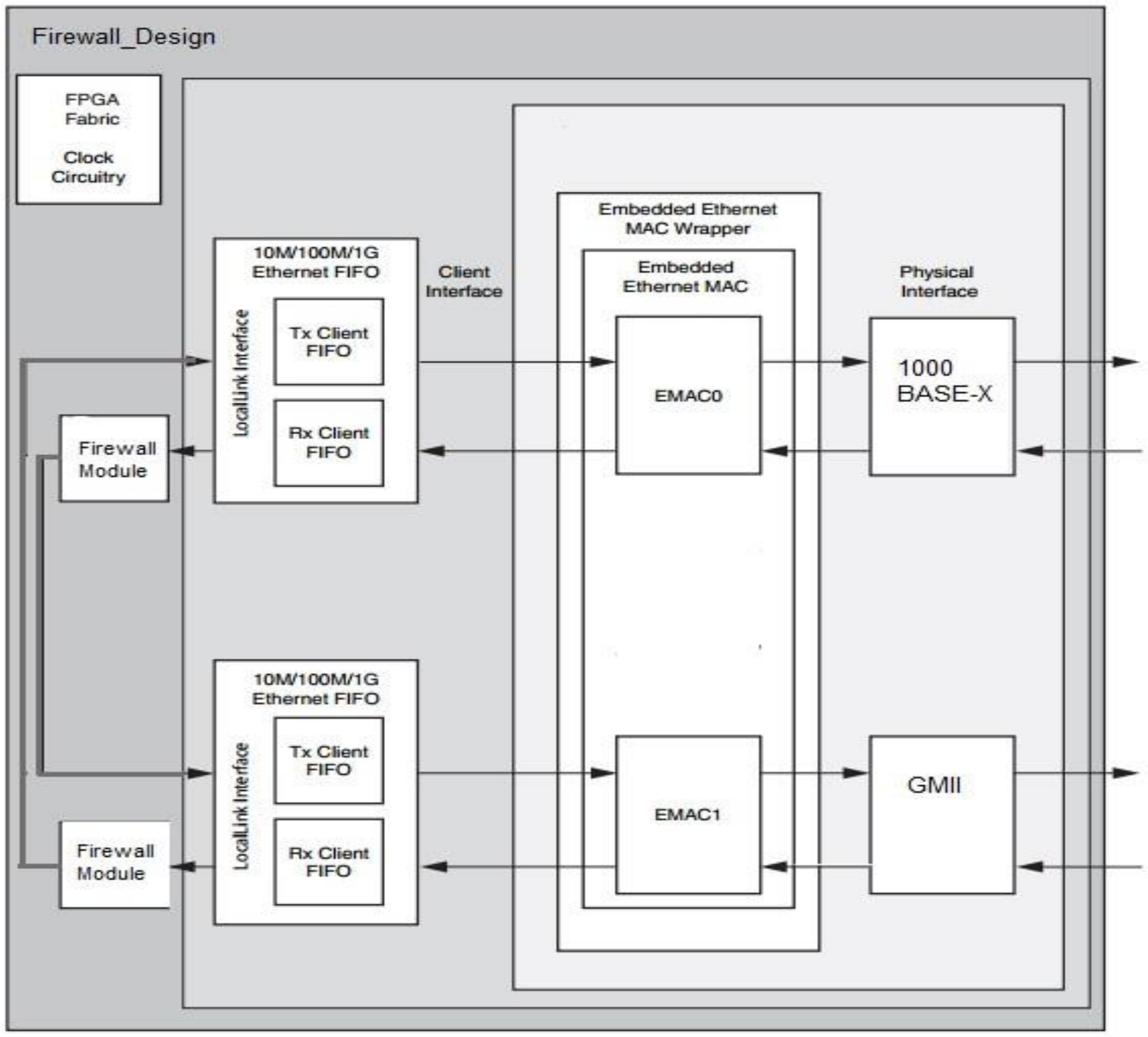


Figure 17. Firewall Design Flow

CHAPTER 5:

IMPLEMENTATION:

5.1 Physical Interface:

An appropriate Physical Interface is provided for each selected EMAC0/EMAC1. This interface connects the physical interface of the Ethernet MAC block to the I/O of the FPGA.

- For GMII/MII, this component contains Input/Output block (IOB) buffers and IOB flipflops.
- For 1000BASE-X PCS/PMA or SGMII, this component instantiates and connects a RocketIO GTP or GTX transceiver

a) 1000 BASE-X:

The 1000BASE-X logic can be optionally provided by the Ethernet MAC. This port is connected to a Switch or Routing matrix, which can contain several ports.

b) GMII(Gigabit Media Independent Interface):

The Media Independent Interface (MII), defined in IEEE 802.3clause 22, is a parallel interface that connects a 10-Mbps and/or 100-Mbps capable MAC to the physical sublayers. The Gigabit Media Independent Interface (GMII), defined in IEEE 802.3clause 35, is an extension of the MII used to connect a 1-Gbps capable MAC to the physical sublayers. MII can be considered a subset of GMII, and as a result, GMII/MII can carry Ethernet traffic at 10 Mbps, 100 Mbps, and 1 Gbps.

5.2 Embedded Ethernet MAC Wrapper:

The Ethernet MAC wrapper file instantiates the full Ethernet MAC primitive. For one or both Ethernet MACs (EMAC0/EMAC1), the following applies:

- All unused input ports on the primitive are tied to the appropriate logic level; all unused output ports are left unconnected.
- The Ethernet MAC attributes are set based on options selected in the CORE Generator software.
- Only used ports are connected to the ports of the wrapper file.

This simplified wrapper should be used as the instantiation template for the Ethernet MAC in customer designs.

5.3 Ethernet MAC Core:

The LogiCORE IP Tri-Mode Ethernet Media Access Controller (TEMAC) solution comprises the 10/100/1000 Mb/s Ethernet MAC, 1 Gb/s Ethernet MAC and the 10/100 Mb/s Ethernet MAC IP core. All cores support half-duplex and full-duplex operation.

5.3.1 Core Overview:

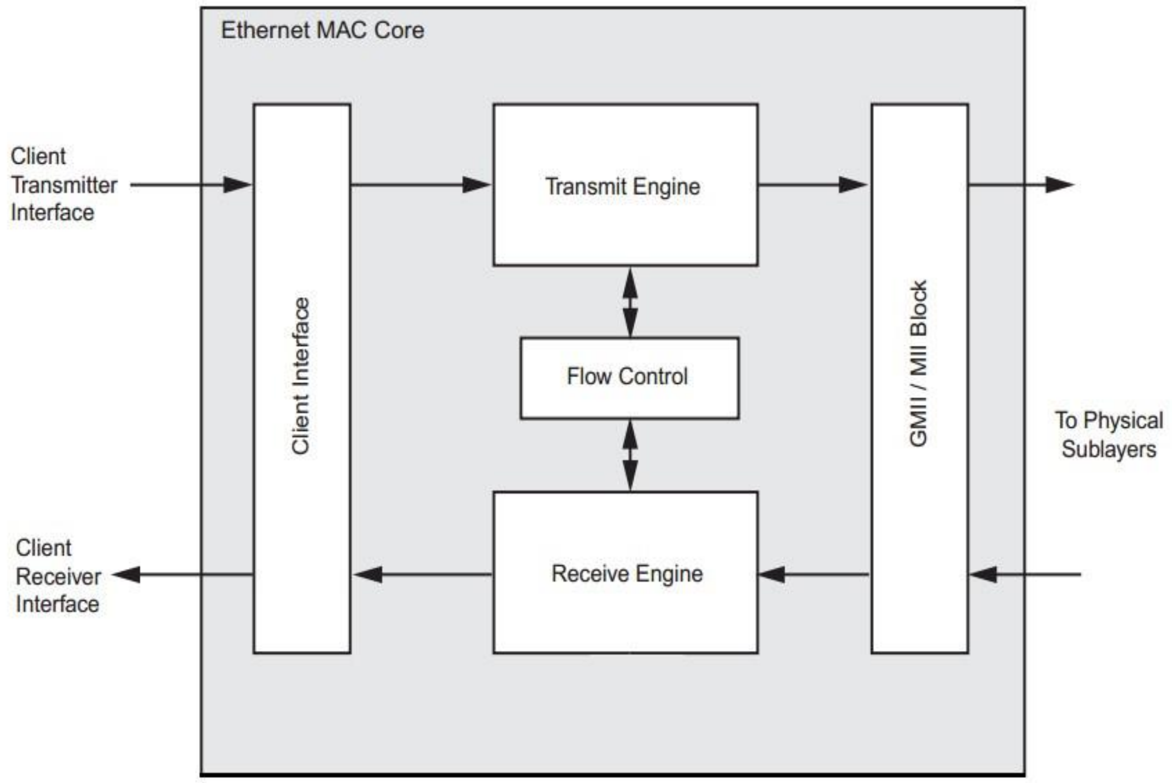


Figure 18. Core Overview

a) Client Interface:

The client interface is designed for maximum flexibility in matching to a client switching logic or network processor interface. The data pathway is 8-bits wide in both the transmit and receive directions.

b) Transmit Engine:

The transmit engine takes data from the client and converts it to GMII format. Preamble and frame check sequence fields are added and the data is padded if necessary. The transmit engine also provides the transmit statistics vector for each packet and transmits the pause frames generated by the flow control module.

c) Receive Engine:

The receive engine takes the data from the GMII/MII interface and checks it for compliance to the IEEE 802.3.Padding fields are removed and the client is presented with the data along with a good or bad frame indicator. The receive engine also provides the receive statistics vector for each received packet.

d) Flow Control:

The MAC can be configured to send pause frames with a programmable pause value and to act on their reception. These two behaviors can be configured asymmetrically.

e) GMII/MII Block:

The GMII/MII interface takes the data from the transmitter and converts it to MII format if the device is operating at speeds under 1 Gb/s. The received data is converted into GMII format. At 1 Gb/s, the data is passed through.

CHAPTER 6:

RESULTS:

6.1 Simulation Results:

Data is given as input in the figure given below:

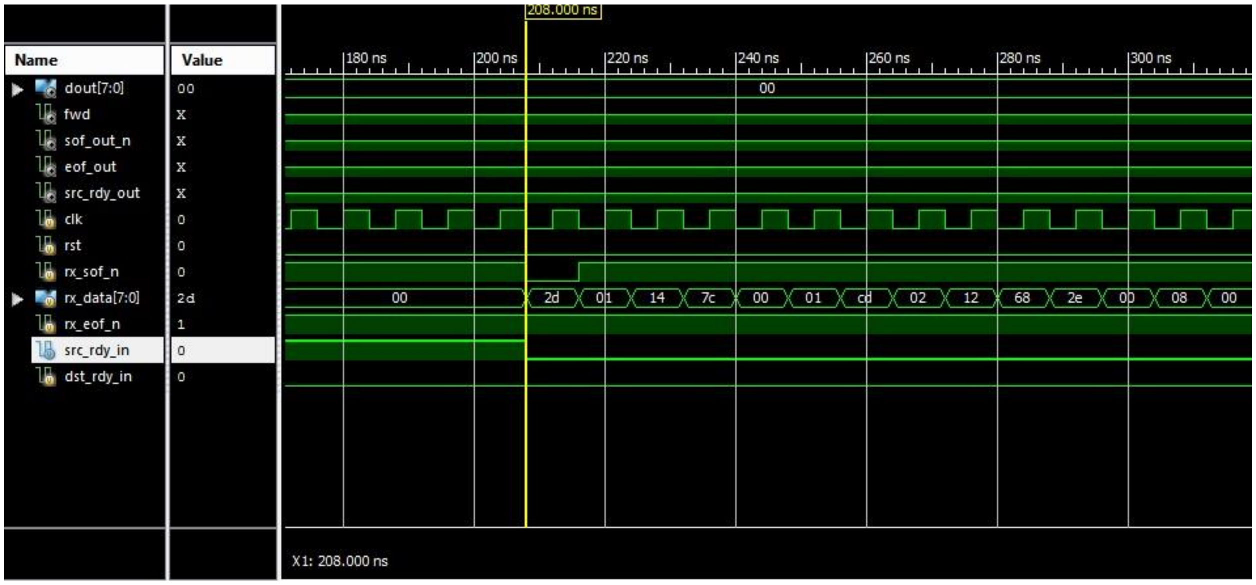


Figure 19. Input Data

Since MAC address is not matched, so the packet is not forwarded as shown:

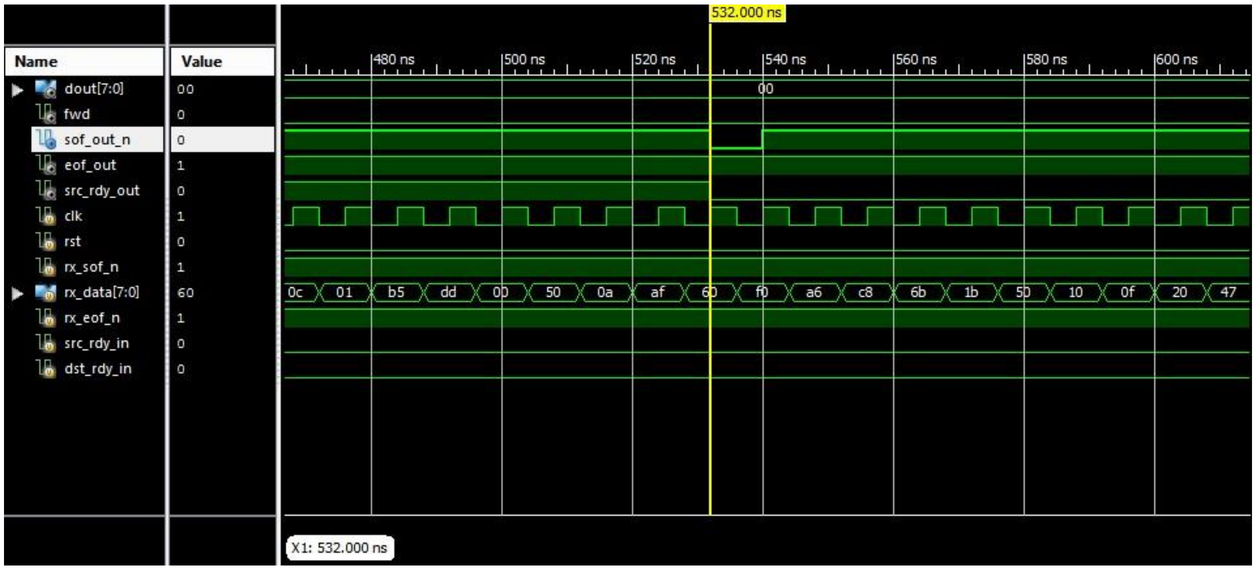


Figure 20. Input Data-Deny

Second Packet is received as input in the figure given below:

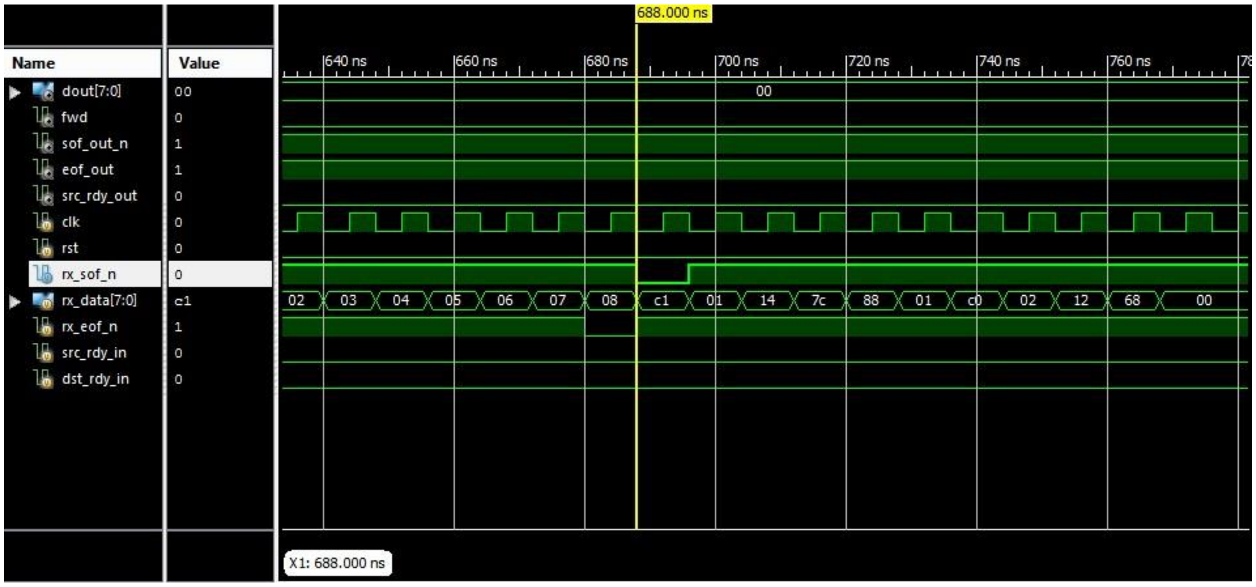


Figure 21. Input Data-2

Since all addresses are matched with pre stored rules, so the packet is forwarded as shown:

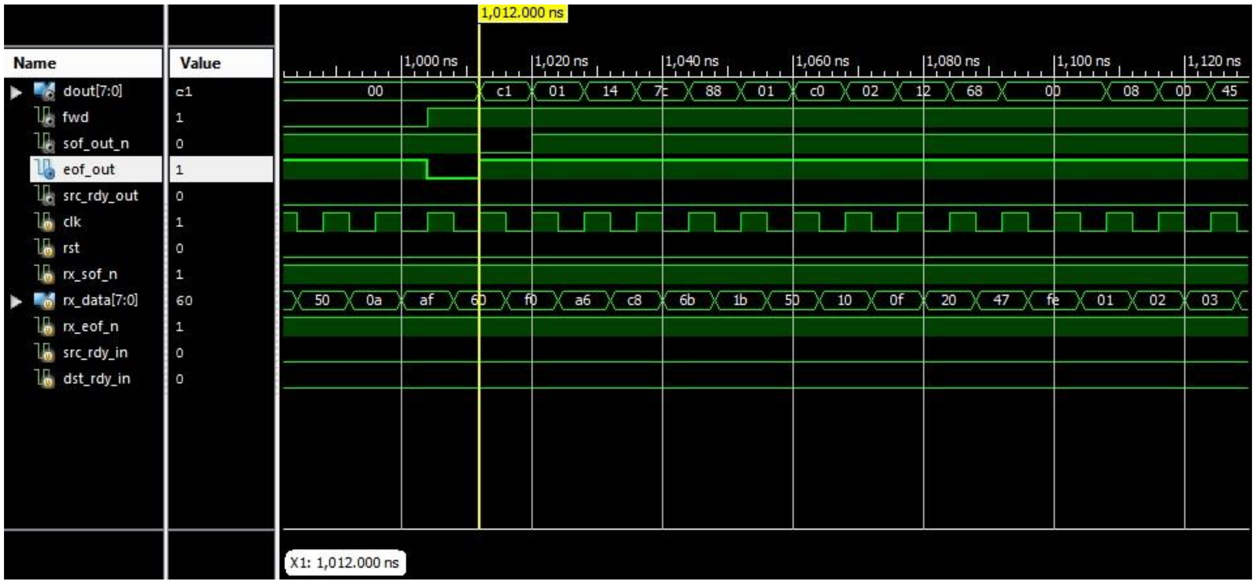
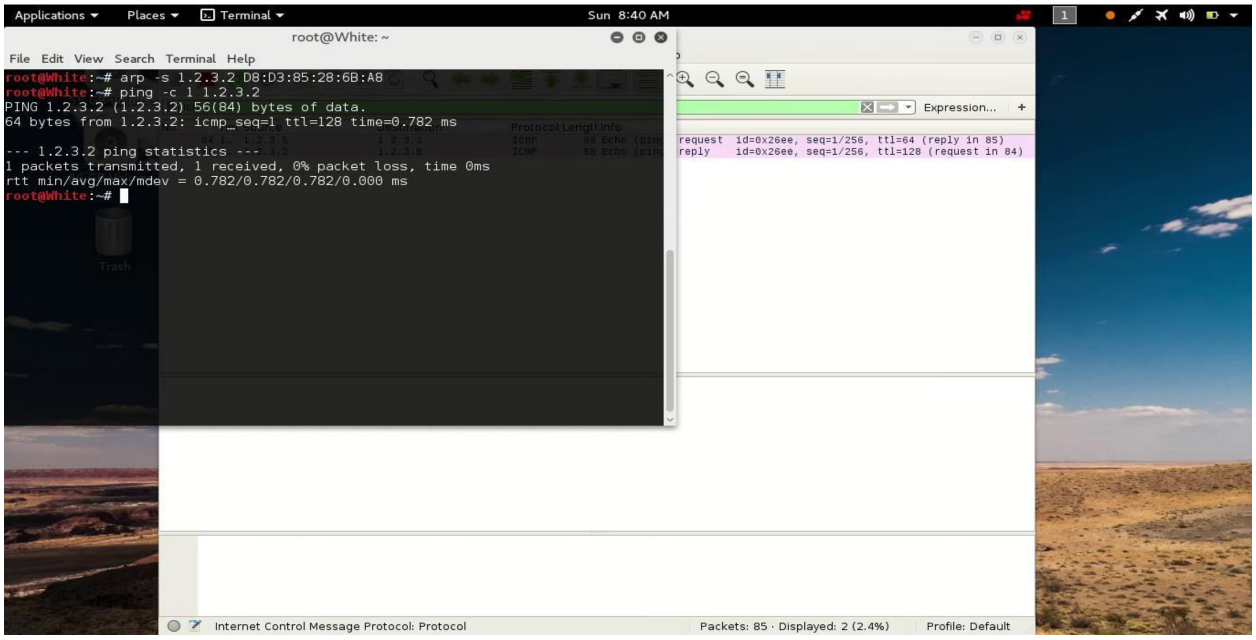


Figure 22. Input Data-Allow

6.2 Implementation Results:

Below is the figure when ping request to allowed IP address:



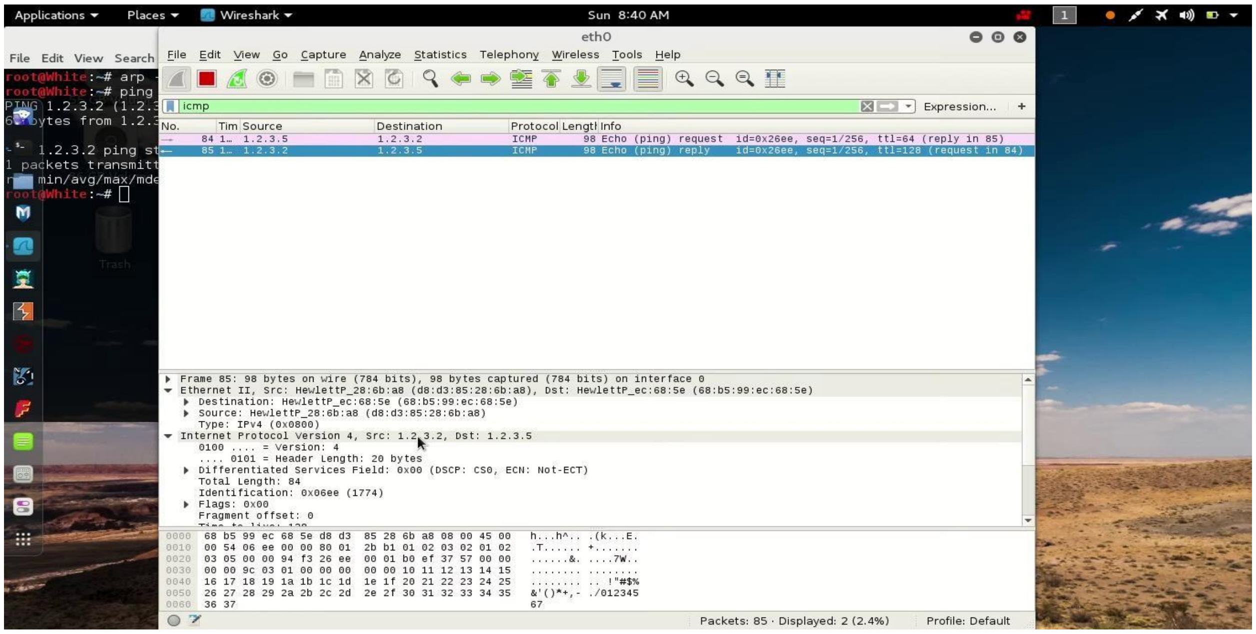
```
root@White:~# arp -s 1.2.3.2 D8:D3:85:28:6B:A8
root@White:~# ping -c 1 1.2.3.2
PING 1.2.3.2 (1.2.3.2) 56(84) bytes of data.
64 bytes from 1.2.3.2: icmp_seq=1 ttl=128 time=0.782 ms

--- 1.2.3.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.782/0.782/0.782/0.000 ms
root@White:~#
```

```
request  id=0x26ee, seq=1/256, ttl=64 (reply in 85)
reply    id=0x26ee, seq=1/256, ttl=128 (request in 84)
```

```
Internet Control Message Protocol: Protocol     Packets: 85 · Displayed: 2 (2.4%)     Profile: Default
```

Figure 23. Ping Request to allowed IP

Below is the Request and Response in Wireshark:


Applications
Places
Wireshark
Sun 8:40 AM
eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
icmp
Expression...

```
root@White:~# arp
root@White:~# ping
PING 1.2.3.2 (1.2.3
6 bytes from 1.2.3

 1.2.3.2 ping st
1 packets transmitt
r  min/avg/max/mde
root@White:~#
```

```
No.   Tim Source    Destination  Protocol Length Info
 84 1…  1.2.3.5   1.2.3.2      ICMP      98 Echo (ping) request  id=0x26ee, seq=1/256, ttl=64 (reply in 85)
 85 1…  1.2.3.2   1.2.3.5      ICMP      98 Echo (ping) reply    id=0x26ee, seq=1/256, ttl=128 (request in 84)
```

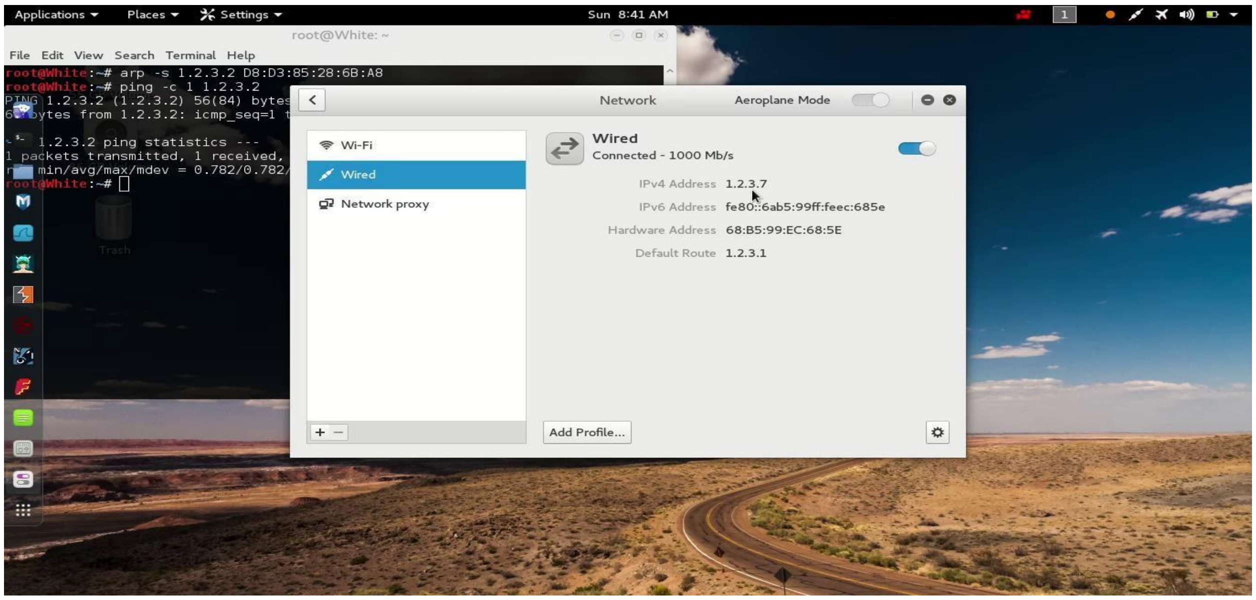
```
▶ Frame 85: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▼ Ethernet II, Src: HewlettP_28:6b:a8 (d8:d3:85:28:6b:a8), Dst: HewlettP_ec:68:5e (68:b5:99:ec:68:5e)
  ▶ Destination: HewlettP_ec:68:5e (68:b5:99:ec:68:5e)
  ▶ Source: HewlettP_28:6b:a8 (d8:d3:85:28:6b:a8)
    Type: IPv4 (0x0800)
▼ Internet Protocol Version 4, Src: 1.2.3.2, Dst: 1.2.3.5
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x06ee (1774)
  ▶ Flags: 0x00
    Fragment offset: 0
```

```
0000  68 b5 99 ec 68 5e d8 d3  85 28 6b a8 08 00 45 00   h...h^.. .(k...E.
0010  00 54 06 ee 00 00 80 01  2b b1 01 02 03 02 01 02   .T...... +.......
0020  03 05 00 00 94 f3 26 ee  00 01 b0 ef 37 57 00 00   ......&. ....7W..
0030  00 00 9c 03 01 00 00 00  00 00 10 11 12 13 14 15   ........ ........
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ........ .. !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67
```

Packets: 85 · Displayed: 2 (2.4%) Profile: Default

Figure 24. Request and Response in Wireshark

IP is changed to 1.2.3.7 which is not allowed so the packet should not go through shown in the figure below



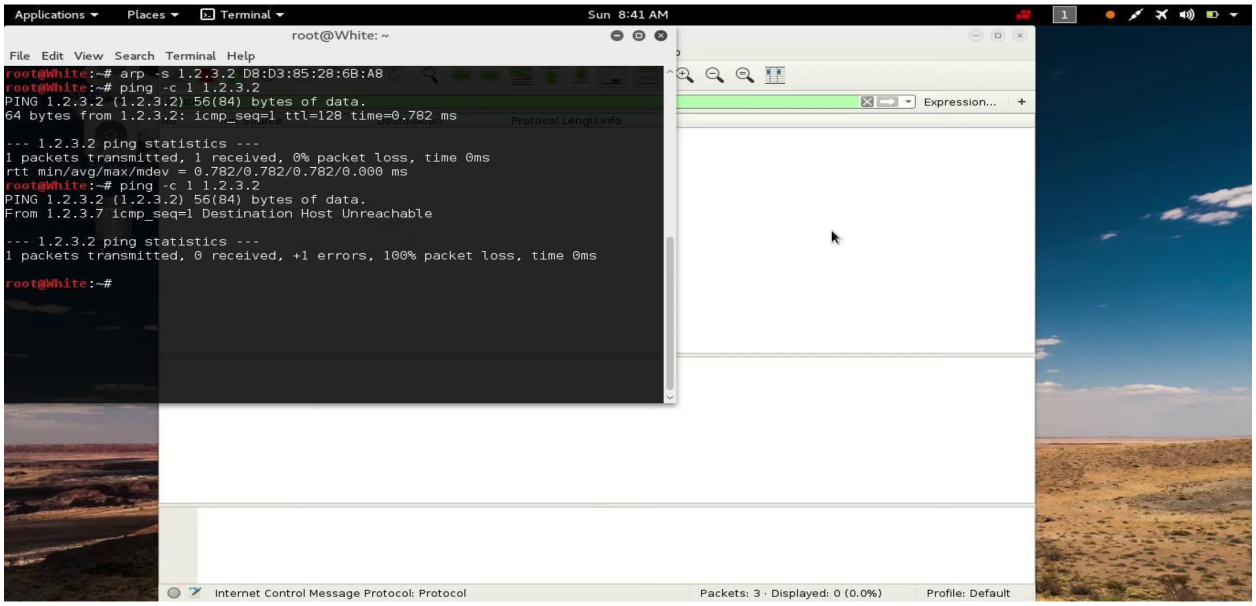
```
root@White:~# arp -s 1.2.3.2 D8:D3:85:28:6B:A8
root@White:~# ping -c 1 1.2.3.2
PING 1.2.3.2 (1.2.3.2) 56(84) bytes
6 bytes from 1.2.3.2: icmp_seq=1 t

 1.2.3.2 ping statistics ---
1 packets transmitted, 1 received,
r  min/avg/max/mdev = 0.782/0.782/
root@White:~#
```

Wired
Connected - 1000 Mb/s

IPv4 Address 1.2.3.7
IPv6 Address fe80::6ab5:99ff:feec:685e
Hardware Address 68:B5:99:EC:68:5E
Default Route 1.2.3.1

Figure 25. Denied IP Address

Now we ping using 1.2.3.7 as source address

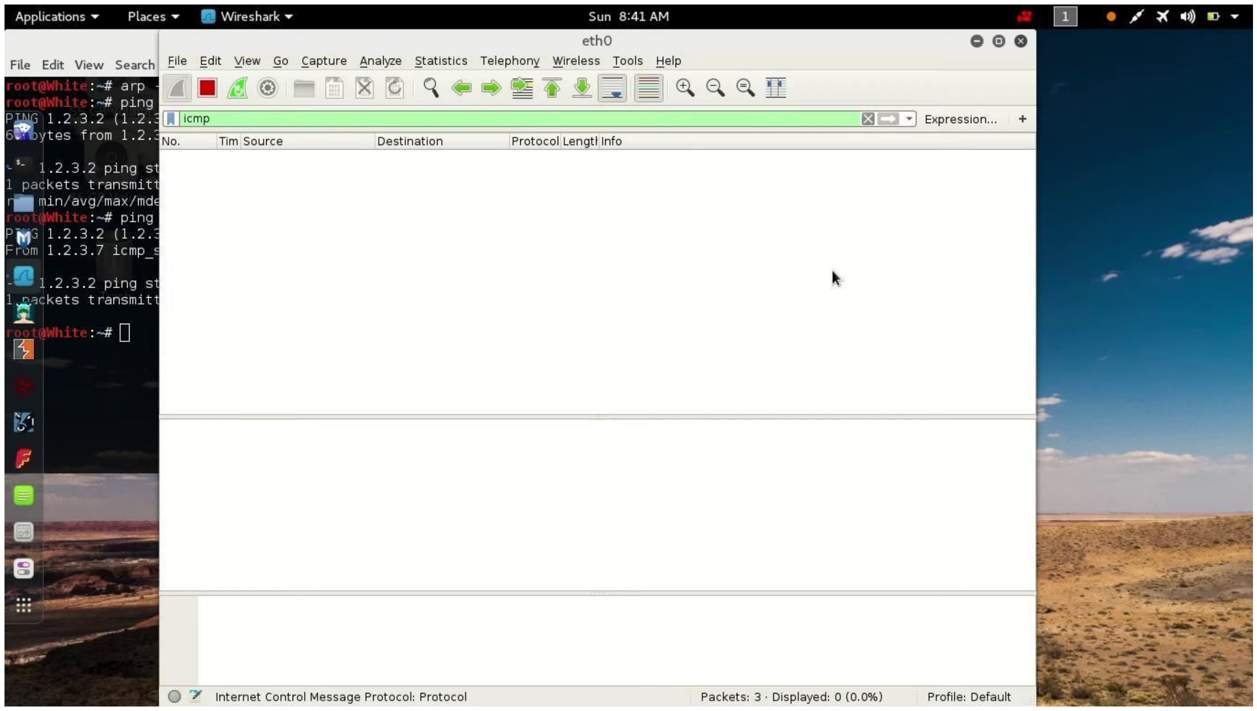


```
root@White:~# arp -s 1.2.3.2 D8:D3:85:28:6B:A8
root@White:~# ping -c 1 1.2.3.2
PING 1.2.3.2 (1.2.3.2) 56(84) bytes of data.
64 bytes from 1.2.3.2: icmp_seq=1 ttl=128 time=0.782 ms

--- 1.2.3.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.782/0.782/0.782/0.000 ms
root@White:~# ping -c 1 1.2.3.2
PING 1.2.3.2 (1.2.3.2) 56(84) bytes of data.
From 1.2.3.7 icmp_seq=1 Destination Host Unreachable

--- 1.2.3.2 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

root@White:~#
```

Applications
Places
Terminal
Sun 8:41 AM
root@White: ~
File Edit View Search Terminal Help
Expression...
Internet Control Message Protocol: Protocol
Packets: 3 · Displayed: 0 (0.0%)
Profile: Default

Figure 26. Denied Source Address

There is no packet displayed in wire shark, which is as expected because IP is not allowed


Applications
Places
Wireshark
Sun 8:41 AM
eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...
No. Tim Source Destination Protocol Lengtl Info

```
root@White:~# arp -
root@White:~# ping
PING 1.2.3.2 (1.2.3
64 bytes from 1.2.3

--- 1.2.3.2 ping st
1 packets transmitt
rtt min/avg/max/mde
root@White:~# ping
PING 1.2.3.2 (1.2.3
From 1.2.3.7 icmp_s

--- 1.2.3.2 ping st
1 packets transmitt

root@White:~#
```

Filter: icmp

Internet Control Message Protocol: Protocol   Packets: 3 · Displayed: 0 (0.0%)   Profile: Default

Figure 27. No Packet in Wireshark

BIBLIOGRAPHY

CHAPTER 7:

BIBLIOGRAPHY:

7.1 List of similar projects done at MCS:

NC Yasir Ahmed, NC Essam Shahid, NC Maheen Zubair, TE-48

Hardware Firewall on FPGA (10G), 2015.

7.2 List of similar projects elsewhere:

- **<u>NETFPGA 10G APPLICATION AWARE ROUTER</u>**

  BY NS Nauman Shahid, NS Asadullah, NS Noman Sharif “” DE-32 Bachelors In Computer Engineering (Eme,Nust) (2014)

  Application Aware Routing which supports the routing of packets based in their OSI 4-7 features i.e. URL and Protocol

- **<u>Designing a Hardware-Accelerated Firewall with Two 10 Gbps Ports</u>**

  BY Viktor Pus,Tomas Dedek (2008)

  Focused on one of these performance-demanding tasks: the packet classification. Implement the stateless firewall because of its straightforward design and clear requirements for its function. Stateless firewalls is a process in which every packet is independently of the previous packets.

- **<u>DESIGN A HARDWARE NETWORK FIREWALL ON FPGA</u>**

  BY Raouf Ajami, Anh Dinh University of Saskatchewan, Canada: (2011)

  Built using Verilog Hardware Description Language (HDL) to speed up the packet processing. A processor based embedded system with real- time operating system was designed to achieve highly customized and on-the-fly configuration change in the firewall. The whole design was implemented and evaluated on an Altera FPGA device.

 <u>FPGA-BASED INTERNET PROTOCOL FIREWALL CHIP</u> BY Ayman, Louis Harik,Rony Ferzli, and Mohammad Fawaz (American University of Beirut, Lebanon) (2000)

Presented the design of a firewall for IP networks using a field-programmable gate array

(FPGA).

7.3 ONLINE HELP:

 www.xilinx.com

- http://www.digilentinc.com/Products/Detail.cfm?Prod=GENESYS
- http://www.asic-world.com/verilog/veritut.html
- www.fpga4fun.com
- http://www.xilinx.com/training/free-video-courses.htm
- http://www.ni.com/swf/presentation/us/labview/lvfpga/
- http://www.xilinx.com/csi/training/how-to-configure-an-fpga.htm
- http://www.xilinx.com/support/index.html/content/xilinx/en/supportNa v/si licon_devices/fpga/virtex-6.html
- http://hardforum.com/showthread.php?t=1074301
- http://www.xilinx.com/ipcenter/coregen/ip_update_install_instructions.ht m
- http://www.xilinx.com/ipcenter/coregen/ip_update_system_requireme nts. htm
- http://www.xilinx.com/products/ipcenter/V5_Embedded_TEMAC_Wrap p er.htm

CHAPTER 8:

PROJECT RESEARCH WORK:

- XILINX SUPPORT DOCUMENTATION, "*Virtex 5 FPGA Configuration User Guide, UG191 (v3.11)”*, October 19, 2012.
- XILINX SUPPORT DOCUMENTATION, “*Virtex 5 FPGA User Guide, UG190* (v5.4)”, March 16, 2012.
- XILINX SUPPORT DOCUMENTATION, “*Virtex 5 FPGA Embedded Trimode Ethernet MAC User Guide, UG194* (v1.10)”, February 14, 2011.
- XILINX SUPPORT DOCUMENTATION, “*Virtex 5 FPGA Embedded Trimode Ethernet MAC Wrapper v1.8, GSG340”*, April 19, 2010.
- XILINX SUPPORT DOCUMENTATION, “*Virtex 5 FPGA Embedded Trimode Ethernet MAC Wrapper v1.8, DS550”*, April 19, 2010.
- Sheila Frankel, *“Demystifying the IPSec Puzzle”*, 3rd ed, Artech House, Inc, 2001.
- Behrouz A. Forouzan, *“Data Communications and Networking”*, 4th ed, McGraw Hill, 2007.
- Steffen, *“Secure Network Communication*”, ZurcherHochshule Winterthur, January 14, 2002.
- IEEE TRANSACTIONS, “*IPSec Implementation on Xilinx Virtex-II Pro FPGA and Its Application”*, Lu, J. ; Appl. Res. Lab., Washington Univ., USA ; Lockwood, J., April 2005