Reconfigurable Architecture for Internet Packet Filter

Navinkumar Gajakumar Patil[1] Hemant Jeevan Magadum[2] Lakshmaiah Alluri[3]
[1]Project Engineer [2,3]Principal Engineer
[1]CDAC, Hyderabad, India [2,3]CDAC, Thiruvananthapuram, India

Abstract— Internet packet filer is one of the most widely used security approach in networking. Now a days the speed of internet is increases so, there is demand for high speed packet classification engine. Most of the available packet classifier engines are based on Ternary Content Addressable Memory (TCAM). This provide the higher throughput at the cost of large resource and high power consumption. Alternative is the linear search algorithm. It is less complex and consume less power but the throughput of this is less. It has most predictable behavior. When there is moderate number of filter rules then linear search algorithm is used. The proposed architecture is modified version of linear filter algorithm. This shows 3 times higher throughput than linear search. Also, it requires small amount of FPGA resources. This packet filter is designed in Verilog and tested on Xilinx Zynq 7000 series FPGA with 30 complex filter rules based on standard five header fields. The proposed architecture works on maximum operating frequency of 215.5 MHz. It supports 1 Gbps internet connection with throughput of 21.55 Mpps (Million packets per second).

Keywords: Network security, firewall, packet filter, linear search, Field Programmable Gate Array (FPGA)

I. INTRODUCTION

Internet is the global system of connected computers [1]. It is not limited for the computers but also many other electronics devices are now connecting to the internet with unique identity ie. Internet of Things (IoT) [2]. The security over the internet is the most challenging job. Network intrusion and denial of services attacks are made to disturb the smooth operation of the network. To maintain confidentiality, integrity of information and to protect the other assets authenticity of the network devices is important [3]. The network security is the wide concept; multiple security techniques used to provide security. Internet packet filtering is the less complex and effective technique for protecting from several security attacks [3].

The packet filter firewall separates the closed secure network from unsecure external network. It can be stateless and stateful firewall [4]. The stateless firewall filter the received packets considering only header fields from the packets. It makes the filtering decision based on predefined access control list. Access control list are categories as white list and black list [5]. White list has default deny policy and vice versa. The white list is the most secure approach as it discard all the packet which does not match with the rules. Stateful packet filter considers the states of the communication. It is slow compared with stateless packet filter.

There are both software and hardware based Packet Filters [6]. Software filter are more flexible but they run on general purpose processor so speed is limited. Dedicated hardware filter ASIC are high speed with the cost of no flexibility. The FPGA based packet filter can be the best choice because it has high speed over the software filter [7] and more flexible than ASIC.

Network layer packet filter takes the decision to allow or drop the packet based on five tuple fields from header such as Source address (SA), Destination address (DA), Source port (SP), Destination port (DP) and Protocol (PR). The different packet classification techniques based on top level approach are exhaustive search, decision tree, decomposition, tuple space [8]. The different packet classification architectures are implemented considering different algorithms aiming high throughput with minimum resources and power required [9-11]. Most of the algorithms based on TCAM they uses full parallel search to provide high throughput over the cost of large resources and power consumption [12-13]. To increase the throughput the memory access to find the matching rule should be reduces. The use of cache memory can reduce the memory access [14] but for high speed connection the required size of cache may be high [8]. The linear search algorithm is the best choice for hardware implementation of filter when there are moderate and constant number of rules [3].

A. Our Contribution

Here we present modified version of linear search algorithm with two stage filter. The Xilinx Zynq 7000 series FPGA is used to implement the design. Which works on maximum frequency of 215.5 MHz.

The reminder of this literature is organised as follows: The section 2 presents the algorithm used for packet filtering. Section 3 Discuss architectural details of the Packet filter. The experimental setup is shown in section 4. The achieved results are presented in section 5. Section 6 show the performance evaluation of filter algorithm. And section 7 make conclusion.

II. PACKET FILTERING ALGORITHM

A. Two stage linear search algorithm

The packet filtering is carried out based on predefined rules. There are no fixed rules for packet filtering, these are subjective to the network environment and security policies [10]. The network admin or the firewall designer is responsible for designing the filtering rules. The rules are defined based on five header fields. In this architecture the rules are defined as white list as it is the most secure kind of approach.

As mentioned earlier linear search algorithm is best suited for implement on FPGA for medium number of rule set, but one drawback of that is latency, it depends on the number of rules. As the number of rules increases the worst case delay increases. The throughput is calculated on worst case delay so it decreases [8].

The two stage linear search is modified version of traditional linear search to increases the throughput of packet filter. Normally in linear search first the filter rules are decided and arranged in descending priority and stored in memory. But in two stage linear search the rule memory is divided into two stage

1) Index memory
   It contains the rule category.
2) Rule memory
   It contains the actual rules based on five header fields.

When parsed packet data receives it first checked with the index memory. If it does not belongs to any rule category it straightaway discarded else the received packet data is transferred to another comparator where the comparator compare this with rules in respective category rule. The header data is searched linearly to find the matching rule and then necessary action is taken.

III. Architecture

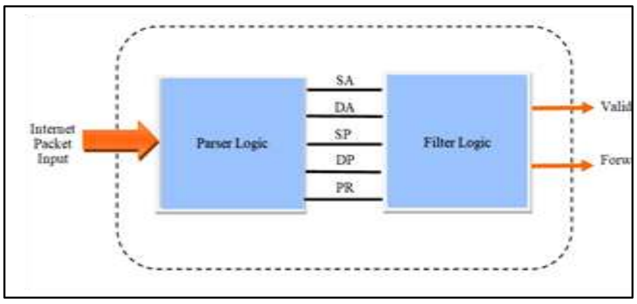


Fig. 1: Block Diagram

This section explains the architecture of packet filter. The top level module consists of two main blocks

1) Parser Logic
2) Filter Logic

A. *Parser Logic:*

The complete received packet is first stored into the first in first out (FIFO) buffer before it goes to the parser logic. The parser logic can be implemented with two approach one is streaming parser and another is non-streaming [15]. Streaming parser start extracting the information as packet data is receiving. It does not store the packets so memory is not required to store the packet but the design of this complicated. Opposite is the non-streaming parser which uses store and forward strategy. It requires the memory and introduce the delay but this is simple to design.

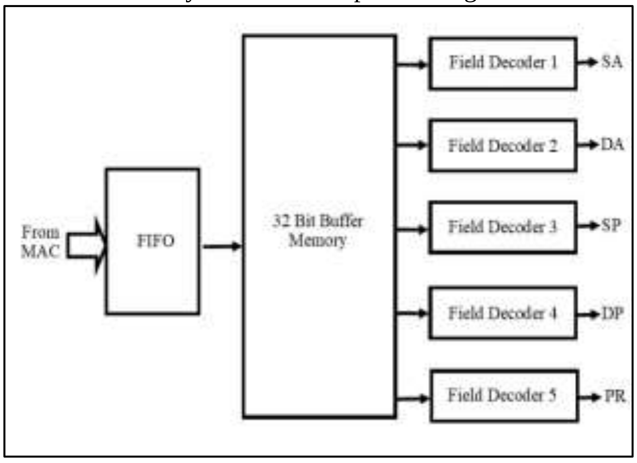


Fig. 2: Header Parser

The data received from FIFO is stored into 32 bit wide memory. At a single shot complete packet is read and given to all field decoder block.

Field decoder extracts the header fields in SA, DA, SP, PD, PR and gives to the filter logic.

B. *Filter Logic:*

The 32 bit source address range is used to categories the rules. Depending on the class of IP address the header information is directed to the respective rule set memory.

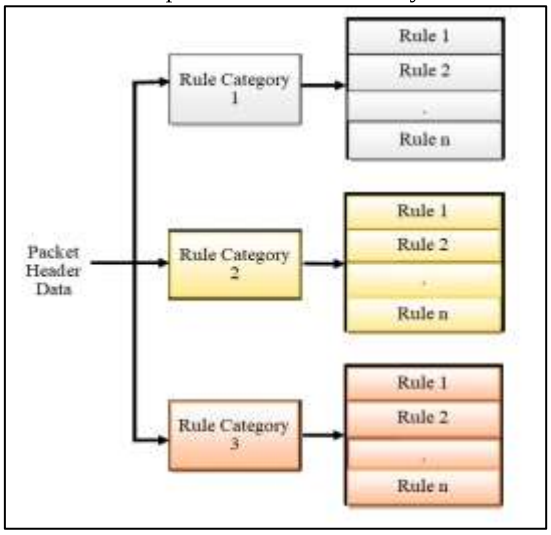


Fig. 3: Two Stage Search Algorithm

Different memory is used to store the rules for each header field. The SA, DA, SP, DP, PR fields of the filter rules are stored in different memories. It allows the parallel access to each field and speed up the performance.

At every clock cycle one location from all the rule memory is accessed and the each header field is compared with the respective header field in rule memory. The result of the comparator is '1' if header field is matches with the rule

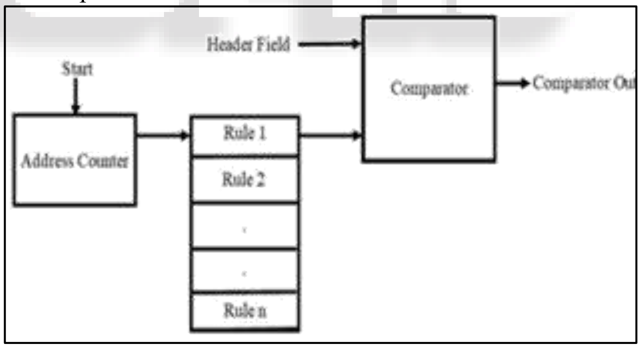


Fig. 4: Linear Search for Filter Rules

else the comparator output is '0'. After that all comparator outputs are AND together to find if the packet is valid based on rule or not. The final output is high only when all header fields of the received packet matches with all the fields stored in the rule memory.

IV. Experimental Setup

The test is carried out by changing only the Source address but it can be verified by changing any five field parameter.

Test condition white list is given in the Table 1. This shows that the received packet matches with the rules are safe and rest of all are unsafe so straight way discard them.

SA	DA	SP	DP	PR
192.168.1.11	192.168.1.10	All	7	TCP

192.168.1.13 - 192.168.1.20	192.168.1.10	All	7	TCP
192.168.1.100	192.168.1.10	All	7	TCP

Table 1: Filter rules sotred in memory at the time of experimentation

For testing, the ZYBO board is connected via USB cable to the laptop. The Ethernet cable is connected to the RJ 45 jack of Board. This is point to point communication so the ZYBO board is initialized with address 192.168.1.10 as static IP address and laptop is configured with manual IP address as Control panel -> Network and sharing center –> Change adaptor setting –> properties. Here the laptop is the source and ZYBO board is the destination.

A. Case 1

The laptop is configure with the IP address 192.168.1.11 and send the message to the ZYBO board using 'Tetra Term' software When the source IP address is matches with the rules in the white list then the 'Valid LED' and 'Forward LED' is turns on. This indicates the received packet is valid so forward it for next process.

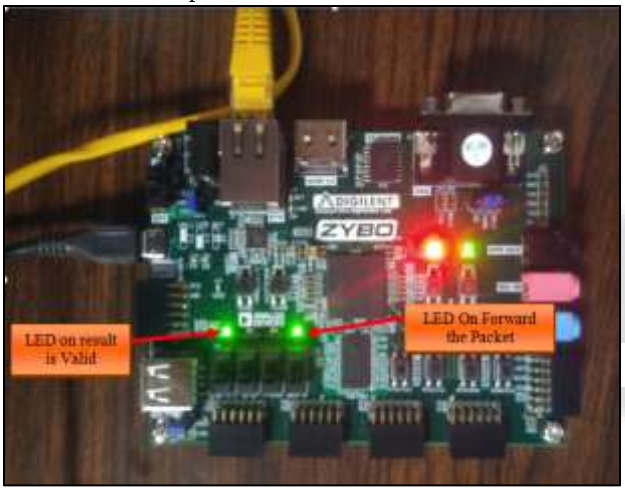


Fig. 5: ZYBO board showing the result is valid and forward LED on

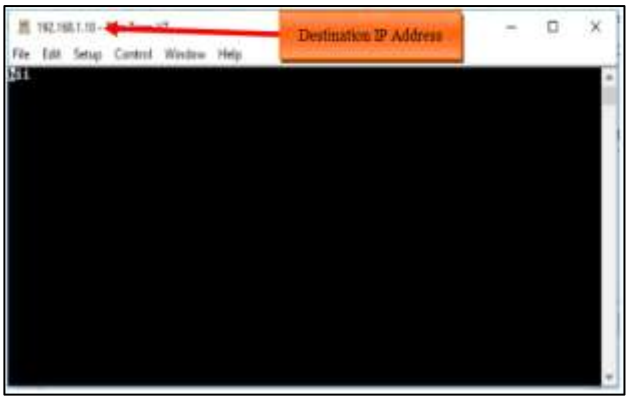
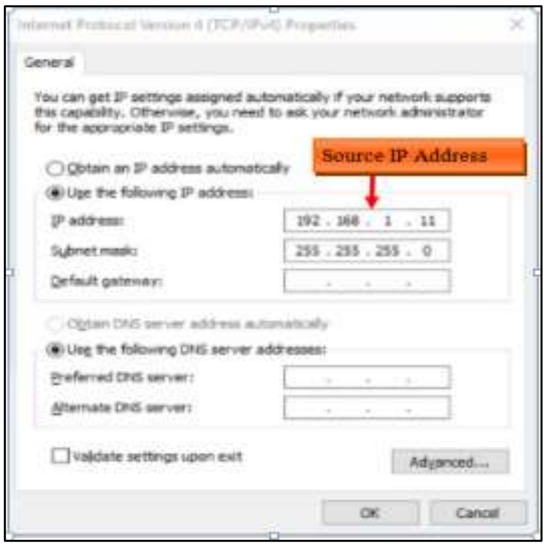


Fig. 6: Sending message via tetra term software to ZYBO board


Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

( ) Obtain an IP address automatically
(●) Use the following IP address:

IP address:	192 . 168 . 1 . 11
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	. . .

( ) Obtain DNS server address automatically
(●) Use the following DNS server addresses:

Preferred DNS server:	. . .
Alternate DNS server:	. . .

☐ Validate settings upon exit

Fig. 7: Configuraing the Laptop with manual IP address which is in the rule set

B. Case 2

Here laptop is configured with IP address 192.168.1.50 as this IP address in not in the rule list, the filter searches for all the rules and when there is no match found then the 'Valid LED' is on to indicate the search is completed and the 'Forward LED' is off; this indicates the received packet does not match with any rule so drop the packet.

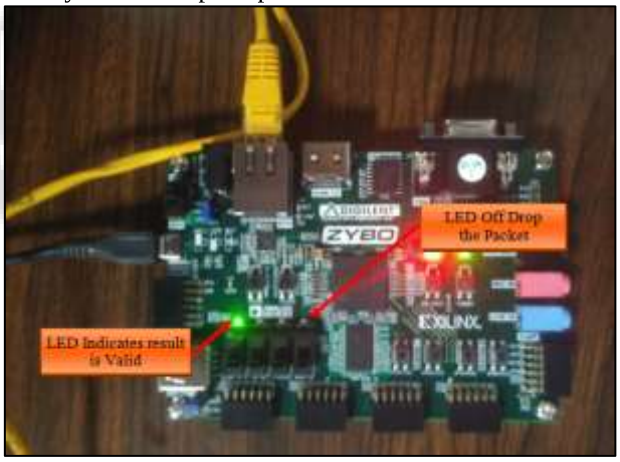
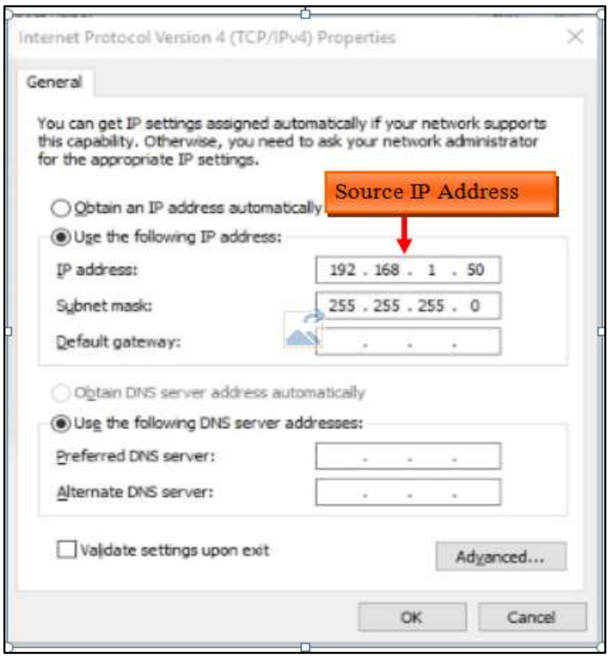


Fig. 8. ZYBO board showing the result is valid and forward LED off ie. the packet doesn't match with the rules

Fig. 9. Sending message via tetra term software to ZYBO board



Internet Protocol Version 4 (TCP/IPv4) Properties

General

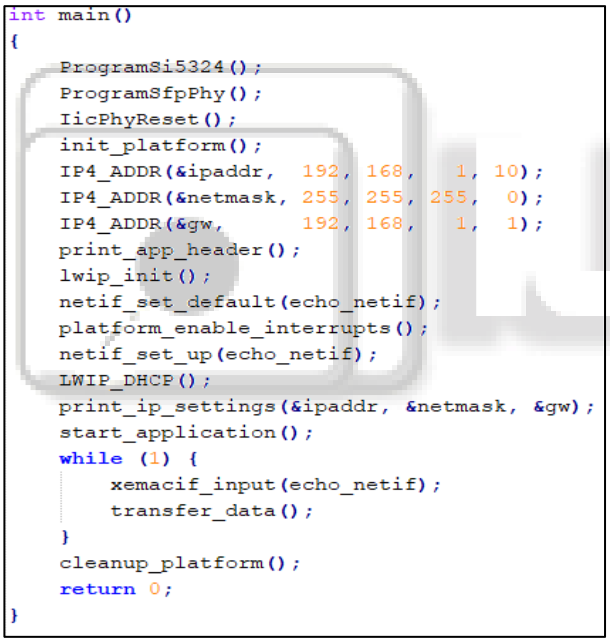
You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

Obtain an IP address automatically
Use the following IP address: [✓]
IP address: 192 . 168 . 1 . 50
Subnet mask: 255 . 255 . 255 . 0
Default gateway: . . .

Obtain DNS server address automatically
Use the following DNS server addresses: [✓]
Preferred DNS server: . . .
Alternate DNS server: . . .

Validate settings upon exit

Fig. 10. Configuraing the Laptop with manual IP address which is not in the rule set



```
int main()
{
    ProgramSi5324();
    ProgramSfpPhy();
    IicPhyReset();
    init_platform();
    IP4_ADDR(&ipaddr,   192, 168,   1, 10);
    IP4_ADDR(&netmask, 255, 255, 255,  0);
    IP4_ADDR(&gw,       192, 168,   1,  1);
    print_app_header();
    lwip_init();
    netif_set_default(echo_netif);
    platform_enable_interrupts();
    netif_set_up(echo_netif);
    LWIP_DHCP();
    print_ip_settings(&ipaddr, &netmask, &gw);
    start_application();
    while (1) {
        xemacif_input(echo_netif);
        transfer_data();
    }
    cleanup_platform();
    return 0;
}
```

Fig. 11: Source Code

V. Results

This architecture is successfully implemented on Xilinx Zynq 7000 series ZYBO board by DIGILENT. Device utilization report in Table 2 shows that this architecture uses very small amount of resources. It uses only one percent of available LUT and Flip-Flops and 8 percent of available input and output port. It has total power dissipation of only 1.693 Watt.

The proposed architecture operates at maximum frequency of 215.5 Mhz which is greater than 125 Mhz. This shows that the architecture is suitable for 1 Gbps internet connection.

As the rules are categorized into three category this shows the memory access per lookup is reduced by factor of three. For 30 rules only 10 memory access are required it reduces the latency.

Resources on FPGA	Used resources	Available resources	% of Utilization
LUT	582	17600	3.31
LUTRAM	62	6000	1.03
FF	664	35200	1.89
IO	3	100	3.00
BUFG	1	32	3.13

Table. 2. Device utilization report performance evaluation

The performance of the packet filter is susceptible to the number of filter rules and how they are categorized. Here the filter rules are categorized in terms of the classes of Source IP address. The performance is based on the assumption that each category has same number of filter rules. The performance of the filter is calculated considering the worst case scenario.

The latency is calculated as: $Latency = \frac{1}{F} * N$

Where F is operating frequency and N highest number rules from all categories

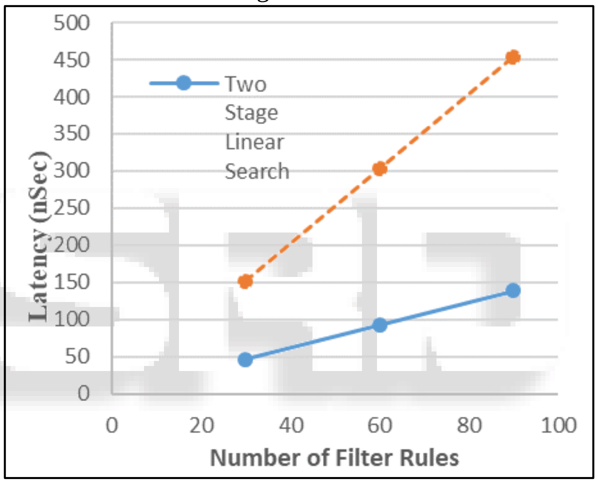


Fig. 12: Latency vs. number of filter rules. Latency is linear with filter rules but for proposed approach it is 1/3 of linear search algorithm.

The Throughput is calculated as: $Throughput = \frac{1}{Latency}$

Where latency is at worst case condition.

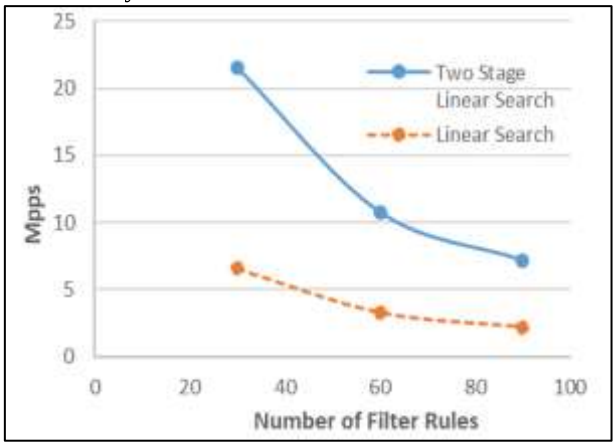


Fig. 13: Throughput vs. number of filter rules. Throughput decreases with filter rules increases but it is always 3 times higher than linear search algorithm.

VI. Conclusion

We proposed the two stage internet packet filter as the modified version of traditional linear search algorithm. The proposed architecture is efficient in terms of area and power. It requires very small amount of FPGA resources. However the proposed method lack in throughput than other Ternary Content Addressable Memory Based Approaches. This architecture is best suited for less number of filter rules, as the number of filter rules increases the throughput decreases. For 30 filter rules with 1 Gbps connection it shows the throughput as the 21.55 Million packets per second.

References

[1] Internet [Online] Available: https://en.wikipedia.org/wiki/Internet
[2] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey,” Comput Netw., vol. 54, no. 15, pp. 2787–2805, 2010.
[3] J. Loinig, J. Wolkerstorfer, and A. Szekely, “Packet filtering in gigabit networks using fpgas,” in Austrochip 2007 Proc. 15th Austrian Workshop on Microelectronics (2007), 2007, pp. 53 – 60.
[4] Stateful-vs-stateless firewalls [Online] Available: https://www.cybrary.it/0p3n/stateful-vs-stateless-firewalls/
[5] IP-blacklist-versus-IP-whitelist [Online] Available: http://www.security-and-privacy-software.com/ip-blacklist-versus-ip-whitelist.html
[6] A. Kayssi, et al, "FPGA-based internet protocol firewall chip ," The 7 IEEE International Conference on Electronics Circuits and Systems ICECS 2000, vol.1. pp. 316-319 , December 2000
[7] H. Chen, Y. Chen, and D. H. Summerville, “A survey on the application of FPGAs for Network infrastructure security,” IEEE Commun. Surveys & Tutorials, Vol. 13, No. 4, pp. 541-561, Fourth Quarter 2011
[8] David E. Taylor, “Survey and taxonomy of packet classification techniques”, ACM Computing Surveys, Vol. 37 No. 3, pp.238-275, September 2005
[9] Jan van Lunteren and Ton Engbersen, “Fast and scalable packet classification,” IEEE Jurnal on selected areas in communication, Vol. 21, No 4, May 2003
[10] Gajanan S. Jedhe, Arun Ramamoorthy and Kuruvilla Varghese, “A scalable high throughput firewall in FPGA,” 16th International Symposium on Field-Programmable Custom Computing Machines, 2008
[11] Thilan Ganegedara, Weirong Jiang and Viktor K. Prasanna, “A scalable and modular architecture for high-performance packet classification” IEEE Transactions on Parallel and Distributed Systems, Vol. 25, No. 5, May 2014
[12] D. Qunfeng, S. Banerjee, J. Wang and D. Agrawal, “Wire speed packet classification without TCAMs: A few more registers (and a bit of logic) are enough.” In SIGMETRICS ’07: Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pages 253–264, June 2007.
[13] S. Hager, F. Winkler, B. Scheuermann, and K. Reinhardt, “MPFC: Massively parallel firewall circuits,” 39th Annual IEEE International Conference on Local Computer Networks, pages 305–313, Sept. 2014.
[14] Nitesh B. Guinde, Roberto Rojas-Cessa and Sotirios G. Ziavras, “Packet classification using rule caching” 4th International Conference on Information, Intelligence, Systems and Applications, July 2013
[15] G. Gibb et al., “Design principles for packet parsers”, in ANCS. IEEE, pp. 13-24, 2013,