High-Throughput Low-Power Multi-Field Pipelined Packet Classification

K Pradeepkumar Reddy [1] , S M K Sukumar Reddy [2]

(MTech in VLSI) [1] , Asst.Prof, E.C.E [2] , V.I.T.S, Proddatur,

Abstract

Packet classification is also called as computer network traffic classification. Packet classification is a process of sorting packets into flows by comparing their headers to a list of rules. A flow is used to decide a packet's priority and the method in which it is processed. The contribution of this paper is a hardware accelerator that can classify up to 406 million packets per second when using rule set containing 20 rules with a peak power consumption of 0.06W when using Vitrex-5 field programmable gate array (FPGA). In this paper hypercuts algorithm is used for packet classification. The architecture used in this paper has two packet classification engines working simultaneously on two different packets in order to improve the throughput.

Keywords—Network traffic classifier, high throughput, hyper cuts, FPGA.

I. INTRODUCTION

As the Internet evolves, it demands next-generation routers to support a variety of network functionalities such as :- Firewall processing, Network Address Translation (NAT), Quality of Service (QoS) differentiation, Virtual Private Networks, Policy Routing,... etc. In order to provide these services, first the router needs to classify the packets into different categories based on a set of predefined rules. Fig. 1 shows the components in a packet classifier.

Network packet is given as input to the classifier module. The management module facilitates the network administrator to configure the set of rules and to associate the rules with the appropriate flows. The classifier compares the input packet with the set of predefined rules and based on the result, it is forwarded to the appropriate flow. At least one rule is associated with each flow and multiple rules can be associated with a flow. In that case, if the input packets matches with any one of the rules mapped to a particular flow, then they are classified into respective flows. If the input packet does not match with any rule, then that packet is dropped or filtered by the classifier.

A packet is considered matching a rule only if it matches with all the fields within that rule. Generally in packet classification, a priority is assigned to all the packets in the same flow.

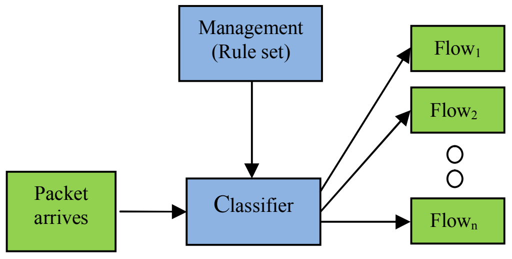

Fig. 1. General structure of packet classifier.

In traditional network applications, packet classification problems usually consider fixed 5-tuple fields: 32-bit source/destination IP addresses 16-bit source/destination port numbers and 8-bit transport layer protocol. This kind of packet classification is called multi-field packet classification. The information presented in this paper is about the design and implementation of packet classification hardware accelerator, which can relieve the network processor from the task of packet classification.

The rest of this paper is organized as follows. Section II explains the architecture of the classifier. Section III explains the architecture of the packet classification engine. Section IV defines the rule set. Section V explains the about the dividing/cutting of the rule set into subgroups. Sections VI and VII explain the decision tree for the rule set and input packets traversing the decision tree respectively. Results are given in section VIII and section IX concludes this paper.

II. ARCHITECTURE OF THE CLASSIFIER

The classifier in Fig. 2 has been implemented with two packet classification engines working in parallel. Packet classification engine-1 reads the input packets from text file-1 and packet classification engine-2 reads the packets from text file-2. Both the packet classification engines work independently of each other and use the same rule set to classify packets. If the input packets match with rule-1 (R1), then they are classified as $Flow_1$, if the input packets match with rule-2 (R2), then they are classified as $Flow_2$ and so on.

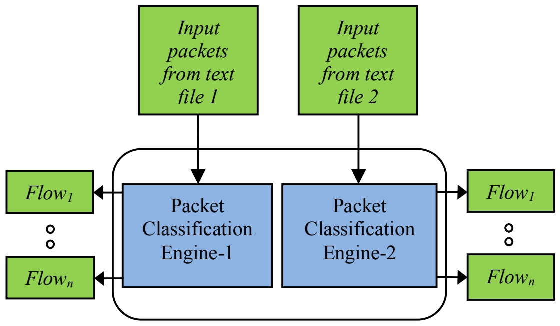
Input packets from text file 1
Input packets from text file 2
Flow1
Flown
Packet Classification Engine-1
Packet Classification Engine-2
Flow1
Flown

Fig. 2. Architecture of the classifier

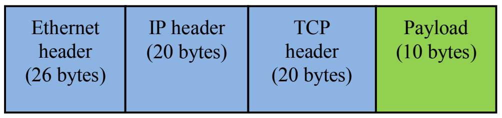

Fig. 3. Input packet format

Two packet classification engines working in parallel are used in this classifier architecture to obtain higher throughput. This architecture is implemented using Verilog hardware description language. In the Verilog Test Fixture, by using the readmemh() library function, the packets which are present in hex format in the text files are read and given as input to the packet classification engines.

Each line in the text file contains one byte. Multiple packets are stored in the text files. In this paper, we have considered 76 bytes as one packet. Fig. 3 shows the packet format used in this paper. It has 26 bytes of Ethernet header, 20 bytes of IP header, 20 bytes of TCP header and 10 bytes of payload. Minimum size IP and TCP headers are used in this packet, it means that the options and padding fields in the IP and TCP headers are not used. Some packets may contain only the headers without the payload. Without the payload, minimum size of the input packet is 66 bytes.

The fields that are used from the IP header are 32-bit Source IP address, 32-bit Destination IP address and 8-bit Protocol. In the TCP header, 16-bit Source and Destination port number fields are used.

III. ARCHITECTURE OF THE PACKET CLASSIFICATION ENGINE

Fig. 4 illustrates the architecture of the packet classification engine. It consists of the following modules: Receiver, Memory, Headers Extraction and Decision tree built using the set of predefined rules. In Fig. 4, all the module blocks are represented in blue color and all the intermediate entities, inputs and outputs are represented in green color.

The receiver module reads the input packets from the text file and validate the input packets. In each clock cycle, receiver module will read one byte of data. For demonstration purposes, here we are providing input from a text file. Though here the receiver module reads input from a text file, it has been designed in such a way that it can integrated with any other source of input. Receiver module uses control signals like SOP (start of packet) and EOP (end of packet) to be in sync with the source of input.

If the input packets are not received in accepted format (protocol), then those packets are discarded. Receiver module will keep track of the number of bytes received. When it receives 76 bytes (one packet), the output signal ‘valid’ goes high and the packet is available on the output port data_out[607:0] of the receiver module. Memory module will store the packets that come from receiver module. Memory module saves one packet in one memory location. When the WE (write enable) signal is enabled, memory module will write the received packet to the memory.

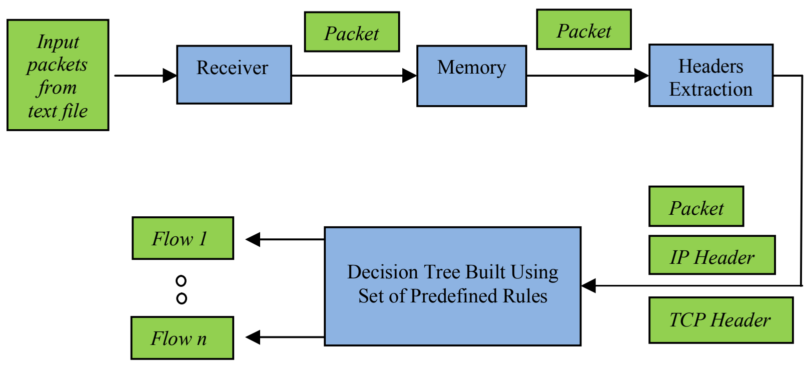

Fig. 4. Architecture of the packet classification engine

Inputs to the headers extraction module are the packets that are saved in memory. Headers extraction module extracts the IP header and TCP header from the packets received and they are given as input to the decision tree. In this paper, HyperCuts algorithm is used to classify the packets. It is a decision tree based algorithm and it works by breaking the rule set into groups, with each group containing a small number of rules suitable for linear search. HyperCuts creates the decision tree by taking a geometric view of the rule set, with each rule considered to be a hypercube in hyperspace. When the input packet is traversing the decision tree, the receiver module will not be idle. It will start receiving the next packet, that is how the pipelining concept has been used in this design and it justifies the term 'pipelined' in the title.

IV. DEFINING RULE SET

Table I shows the rule set containing 20 rules. Rules are defined based on the five header fields; Source IP address, Destination IP address and protocol are taken from IP header of the network packet; and the header fields Source Port and Destination Port are taken from TCP header of the network packet. All the input packets matching Rule ID (R1) are classified as Flow 1 and all the input packets matching Rule ID (R2) are classified as Flow 2 and so on.

In table I, 'X' represents don't care, it can be either '0' or '1'. In rule-1, the source IP address value is 00XX, it means that the least significant four bits of source IP address in input packet can be one of the following values 0000, 0001, 0010, and 0011. Most of the rules in table I are defined using only three header fields and only few rules are defined using all the five header fields. The symbol '-' in the table I indicates that the particular header field is not used in defining that particular rule. Rule numbers R1 to R20 indicate the Rule IDs and the Flow numbers Flow1 to Flow20 indicate the Flow Ids.

For defining the rules, only few bits of the header fields are used. Number of bits used in each header field is indicated in the table I. Least significant four bits of source and destination IP addresses are used. Least significant five bits of the source port number header field and least significant four bits of the destination port number header field are used. The values in Protocol header field are either TCP or UDP. The numeric value for TCP is 6 and the numeric value for UDP is 17. The rules defined here are a combination of range matching and exact matching. The values in source and destination IP address header fields are example of range matching.

TABLE I: Rule Set Containing 20 Rules

Rule ID	Source IP Address [3:0]	Destination IP Address [3:0]	Source Port [4:0]	Destination Port [3:0]	Protocol	Classification (Flow ID)
R1	00XX	00XX	0001	0101	TCP	Flow 1
R2	00XX	00XX	0001	1010	UDP	Flow 2
R3	00XX	00XX	0010	-	-	Flow 3
R4	00XX	01XX	0011	1011	UDP	Flow 4
R5	00XX	01XX	0011	1100	TCP	Flow 5
R6	00XX	01XX	0100	-	-	Flow 6
R7	00XX	10XX	0101	-	-	Flow 7
R8	00XX	11XX	0110	-	-	Flow 8
R9	01XX	00XX	0111	-	-	Flow 9
R10	01XX	01XX	1000	-	-	Flow 10
R11	01XX	10XX	1001	-	-	Flow 11
R12	01XX	11XX	1010	-	-	Flow 12
R13	10XX	00XX	1011	-	-	Flow 13
R14	10XX	01XX	1100	-	-	Flow 14
R15	10XX	10XX	1101	-	-	Flow 15
R16	10XX	11XX	1110	-	-	Flow 16
R17	11XX	00XX	1111	-	-	Flow 17
R18	11XX	01XX	10000	-	-	Flow 18
R19	11XX	10XX	10001	-	-	Flow 19
R20	11XX	11XX	10010	-	-	Flow 20

It means that the values of source and destination IP address in the input packet must be in the range specified in the rule to match that particular rule. The values in header fields: source port, destination port and protocol are examples of exact matching. It means that the values of source port, destination port and protocol fields in the input packet must be the exact value specified in the rule to match that particular rule.

V. CUTTING THE RULE SET

Hyper cuts algorithm works by recursively breaking the rule set into small groups, until the rules in each subgroup are less than a predetermined number. Initially the entire rule set present at root node is divided into sub regions using the source IP address and destination IP address. Least significant four bits of source and destination IP addresses are used for cutting the rule set at root node.

Entire rule set is represented in two dimensional spaces as shown in Fig. 5 using the selected header fields. All the rules inside the rectangular box are considered as one group. The rules R1 to R3 form a subgroup; rules R4 to R6 form a subgroup and remaining subgroups contain only one rule. The subgroups containing more than two rules are divided further using the least significant bits of source port header field. The recursive process of cutting the rule set stops at this step as each subgroup contain rules less than or equal to two. Cutting the rule set at internal nodes is represented in Fig. 6. We can see from Fig. 6 that each subgroup contain rules less than or equal to two.

VI. DECISION TREE FOR THE RULE SET

The concept of cutting the rule set is illustrated in this section using the decision tree in Fig. 7. All the input packets start the tree traversal at the root node. Based on the SIP and DIP header fields of the input packet, it is passed on to one of the child nodes of the root node. Then at internal nodes of the tree, further traversing decision is taken based on the source port header field. At leaf nodes, based on the conditions satisfied by the input packet, it is classified into one of the flows.

More number of cuts has been made to rules at root node to keep the decision tree as shallow as possible. In shallow trees, the traversal time is minimal. Four cuts have been made to SIP and DIP header fields each at root node. So the result is 16 subgroups and each subgroup becomes a child node to the root node. As we can see in Fig. 7, there are 16 child nodes to the root node. One cut is made to source port header field at nodes 1 and 2. It results in two subgroups at nodes 1 and 2. One subgroup contains two rules and other subgroup contains one rule. At the end of tree traversing, the input packet is classified into one of the flows.

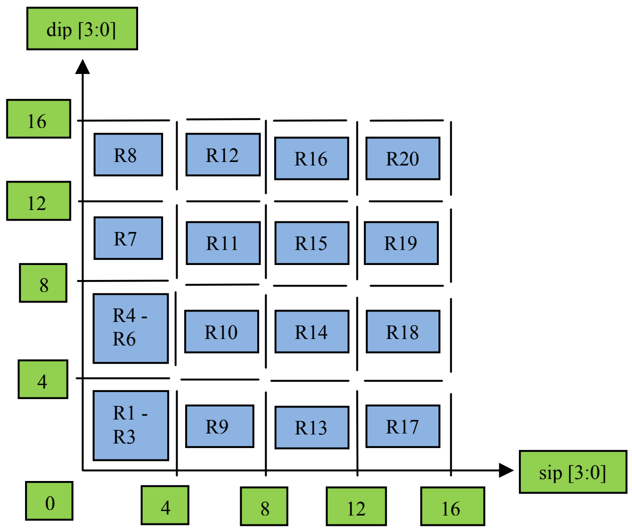

Fig. 5. Cutting the rule set at root node using SIP and DIP header fields

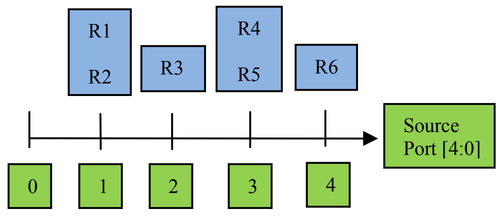

Fig. 6. Cutting the rule set at internal nodes using source port header field.

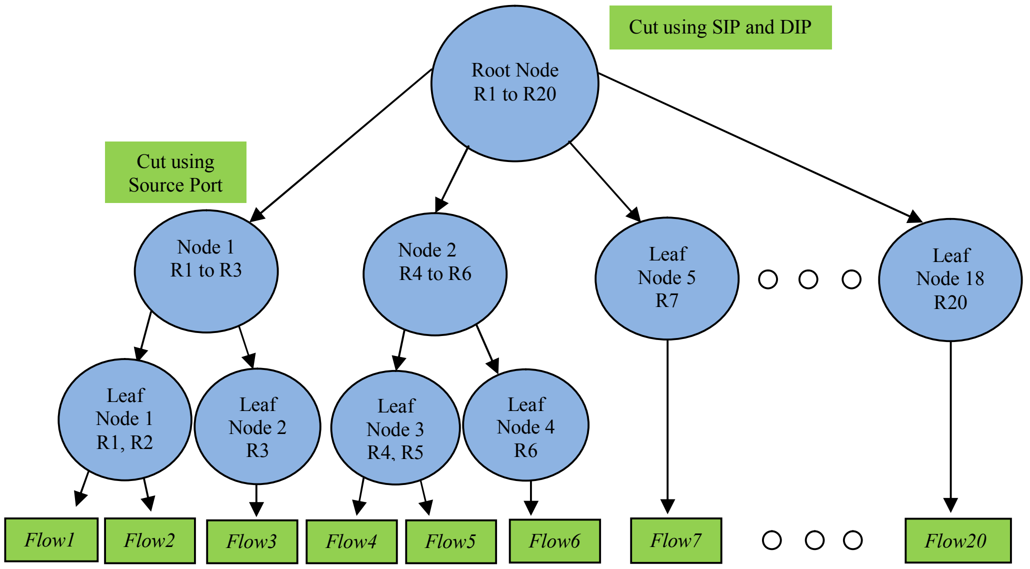

Fig. 7. Decision tree for the rule set containing 20 rules.

VII. INPUT PACKETS TRAVERSING THE DECISION TREE

Fig. 8 shows the input packets traversing the decision tree. Each input packet is represented in hexadecimal notation in a different colour. In each input packet source port header field is highlighted with yellow colour. In the input packets, destination port is immediately after the source port and source IP and destination IP addresses are immediately before the source port. The traversing paths of the input packets are represented by the arrows of same colour as used for the packets. After traversing the decision tree, packet1, packet2, packet3 are categorized as flow-3, flow-5 and flow-20 respectively. The curved arrows are used to point to the nodes which are cut using source port. SIP and DIP header fields of the input packet are used at root node to determine the child node to which the input packet has to traverse. At internal nodes, source port of the input packet is used to decide its traversing path.

```
000475C828E5000102FA70AA08004500002E0001000004068BAA0A0C0E050C060709001100146001000000000000005002000
```

```
000475C828E5000102FA70AA08004500002E0001000004068BAA0A0C0E050C060709001100146001000000000000065003000
```

```
8BAA0A0C0E050C06070904470014600100000000000005000000A30590000FF00AA55C3BEC704DD7B000D75C828ED0012
```

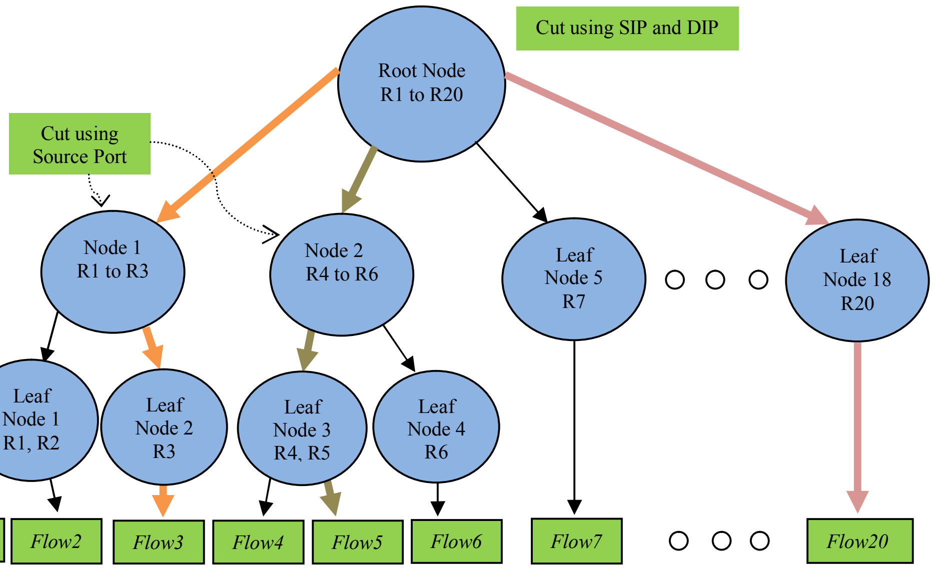

Fig. 8. Three input packets traversing the decision tree (Each input packet is represented in a different colour)

VIII. PERFORMANCE RESULTS

The Virtex-5 FPGA implementation of this architecture achieves the maximum clock speed of 406MHz, giving it a maximum throughput of 406MPPS (Million packets per second).Virtex-6 FPGA implementation of this architecture achieves maximum frequency as 469MHz, allowing it to reach line speeds of up to 469MPPS. As shown in table II the performance of this classifier is further improved by using Virtex-7 FPGA.

Table III shows the performance comparison of classifiers. The classifier from previous work in [3] also implements a modified version of the hyper cuts packet classification algorithm that can classify packets at speeds of up to 169 MPPS. The paper in [4] implements a decision tree-based, dual pipeline architecture that can classify 250 MPPS. The classifier from [6] was implemented on a larger Stratix EP4SGX530 FPGA manufactured using 40 nm process technology, which Altera claim is 35% faster than other devices such as Virtex-5 FPGAs. The throughput of this classifier is 100Mpps. It can be seen from table III that the new classifier outperforms all other classifiers in terms of speed.

IX. CONCLUSION

This classifier can classify up to 406 million packets per second (MPPS) by using virtex-5 FPGA and consumed only 0.06W power. The throughput of this classifier is high when compared to other FPGA based classifiers. This classifier used a modified version of hyper cuts algorithm. The modifications include changing the cutting scheme and index generation.

REFERENCES

[1] D. E. Taylor and J. S. Turner, "Scalable packet classi cation using distributed crossproducting of eld labels," in Proc. IEEE Int. Conf. Comput. Commun., Mar. 2005, pp. 269–280.
[2] A.Kennedy, Z. Liu, X. Wang, and B. Liu, "Multi-engine packet classi cation hardware accelerator," in Proc. 19th Int. Conf. Comput. Commun. Netw., Aug. 2009, pp. 1–6.W.
[3] Jiang and V. K. Prasanna, "Large-scale wire-speed packet classi cation on FPGAs," in Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays, Feb. 2009, pp. 219–218.
[4] S. Singh, F. Baboescu, G. Varghese, and J. Wang, "Packet classi cation using multidimensional cutting," in Proc. ACM Special Interest Group Data Commun. Conf., Aug. 2003, pp. 213–224.
[5] T. Zhang, Y. Wang, L. Zhang, and Y. Yang, "High throughput architecture for packet classi cation using FPGA," in Proc. 5th ACM/IEEE Symp. Archit. Netw. Commun. Syst., Oct. 2009, pp. 62–63.

TABLE II
FPGA RESOURCE UTILIZATION FOR CLASSIFIER

Device	**Maximum Frequency MHz**	**Logic Utilization**	
		Number of Slice Registers	**Number of Slice LUTs**
Virtex 5 xc5vfx200t	406	9666/122880 (7%)	5551/122880 (4%)
Virtex 6 xc6vcx240t	469	9666/301440 (3%)	5120/150720 (3%)
Virtex 7 xc7vx330t	502	9666/408000 (2%)	5120/204000 (2%)

TABLE III
PERFORMANCE COMPARISON OF CLASSIFIERS

Approach	**Device**	**Speed (MPPS)**
Ultra-wide [2]	Stratix III	169
Pipelining [3]	Virtex 5	250
HiCuts [5]	Stratix IV	100
New classifier	Virtex 5	406