Memory Efficient Packet Classification on FPGA

Prof. Deepak Kisan Thakkar
Dept of Electronics and Telecommunication
JESITMR, Nashik

Abstract*- Packet classification is basically used to sorting the packets using different header fields namely source address, destination address, source port number, destination port number etc. comparing with the rules in the classifier. Packet classification is mostly used in Networking and its great challenge to develop scalable solutions for advance packet classification. In this paper, basically two packet classification algorithms, HiCuts and HyperCuts algorithms are simulated on latest software Xilinx Vivado 2016.1 and implemented on hardware that is Genesys-2 Kintex 7 FPGA board. Implementation results shows that HyperCuts algorithms is always better than HiCuts algorithm with many parameters such as depth of decision tree, search speed, Execution time and most important less memory requirements.*

Keywords- FPGA, HiCuts, HyperCuts, Packet Classification

I. INTRODUCTION

Routers provides different network services such as Firewall, Quality of Services (QoS), Virtual private network, policy routing and many more services. To provide these services, routers must be classify packets using sets of rules and the process is called as “Packet Classification” and collection of rules is called as “Classifier”. Basically packet consists of packet header and payload. Packet having different header fields such as source address, destination address, source port number, destination port number and protocol field and payload having information which either voice, data etc. In early days, packet classify using 2- fields but now a days many fields are used to classify packets which are called as multi-field packet classification. In this paper, decision tree based algorithms that are HiCuts and HyperCuts algorithms implemented on Genesys-2 board having Kintex-7 FPGA and simulated on Vivado 2016.1 software.

The paper consists of different sections. Section I is introduction about the work and Section II shows the different research done on FPGA based Packet Classification. Section III has details explanation of HiCuts and HyperCuts algorithm with example. Section IV describes implementation of algorithms and Section V gives the details results after implementation. Finally in last section conclusion of work produces.

II. RELATED WORK

In recent years, variety of packet classification schemes has been proposed to solve the general problem of

Multi-field packet classification. Multi-field packet classification is evolved from traditional 5-tuple packet classification by simply adding packet header fields. Most of researchers work on the implementation of packet classification on FPGA.

Taylor [1] proposes survey and taxonomy of packet classification techniques; in different packet classification technique they combined algorithmic and architectural approach to solve packet classification problems. Also by using taxonomy based high level approach they provide recent solution to the problems as well as various packet classification algorithms by using particular example are easily understand.

Gupta and McKeown [2] introduced algorithm, hierarchical intelligent cutting that is HiCuts algorithm. HiCuts performs well on any classifier available and builds the decision tree from the particular classifier. Every time packet arrives, it can traverse decision tree and find out leaf node which containing small number of rules. A linear search is used to find out best matching rule among the different rules. This algorithm is suitable for hardware implementation as well as it can implement by using software also.

Singh et al. [3] proposed new algorithm, HyperCuts algorithm in which we have to take multiple cuts simultaneously. HyperCuts algorithm is decision tree based algorithm and it is better than HiCuts algorithm in most of the ways. In HyperCuts algorithm the depth of decision tree is reduced upto 1, it requires less memory, optimized for speed as well as fast updates as compared to the HiCuts algorithms. Kennedy and Wang [4] implemented HyperCuts algorithms on Stratix-III FPGA. For packet classification they used networking equipment that for sorting the packets into flows by comparing headers to rules and find out best matching rules among the list of rules. By using this they can classify up to 433 million packets per second which containing tens of thousands of rules set and peak power consumption is 9.03 W. Hardware accelerator uses modified version of HyperCuts algorithm which reduces amount of memory needed for larger rule sets so that which is best fit in the on chip memory of FPGA.

Jang and Prasanna [5] implemented decision tree based algorithm by using linear multi-pipeline architecture on virtex-5 FPGA for multi-field packet classification. They considered advance packet fields problems which consisting of more than 5-tuple header fields would classified. By using different techniques for decision tree based packet classification they reduces memory requirements such that 10K 5-tuple rules or 1K 12-tuple rules could fit into on-chip memory of single FPGA. Simulation and FPGA implementation shows the best results among the different solutions.

From the literature review, it is observed that there is a need to develop a system which classify large packet and provides improved performance with less memory requirement. To achieve this, system mainly uses two algorithms named as HiCuts and HyperCuts. The work is more effective by using Genesys-2 Kintex-7 FPGA board as well as latest software Xilinx Vivado2016.1 uses for simulation of design.

III. DECISION TREE BASED PACKET CLASSIFICATION ALGORITHMS

Next generation packet classification is extension of traditional 5-tuple packet classification which can studied in past decade. Here mainly decision tree based packet classification algorithms that are HiCuts and HyperCuts algorithms are explained in detail. The simplest way to match the header fields of packet to rules is linear search through one rule at a time starting from highest priority rule to lowest priority rule. But for this process large amount of processing time is required and which is difficult to classify packets at the speeds required for core of network. This large amount of processing time is reduced by HyperCuts algorithms. HyperCuts packet classification algorithms is mainly works by breaking the larger rule sets into groups and each group consists of small number of rules which is suitable for linear search. Each group of rules stored in leaf node of decision tree which is suitable for find best matching rule through linear search.

Consider the example of Hicuts and Hypercuts decision trees for 2-fields that are Source Port and Destination Port fields. The rules which can be used to build decision trees are R1-R7. Figure 2 and Figure 3 shows the HiCuts and HyperCuts decision tree respectively builds by using rule classifier. In HiCuts algorithms only one cut taken along the any field while in HyperCuts algorithms simultaneously cutting takes place along all fields.

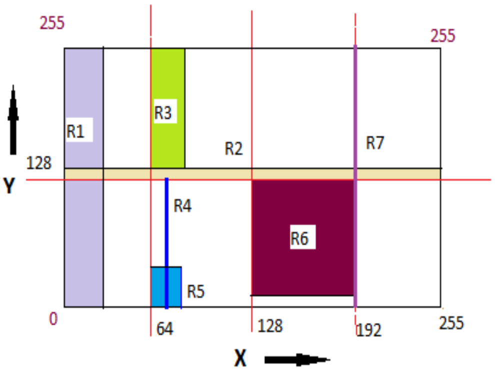


Figure.1: An example of 7-rule classifiers in 2-dimensions.

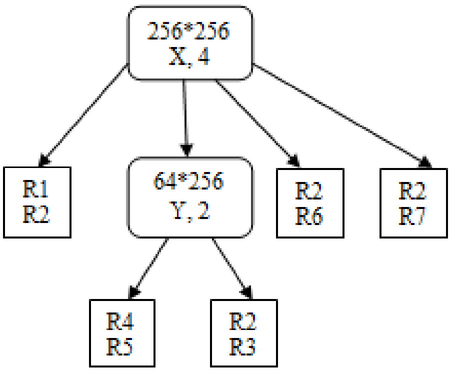


Figure.2: A possible tree (For HiCuts) with binth=2 for the example classifier in figure.1.

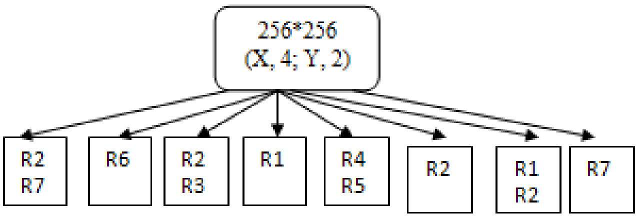


Figure.3. A possible tree for HyperCuts) with binth=2 for the example classifier in figure.1.

IV. IMPLEMENTATION

The architecture of packet classification engine has two modules as shown in Figure 4. The first module is tree traverser that is used to traverse decision tree until the empty node is reached means that there is no matching rule or leaf node is reached. Leaf node is reached means that tree traverser passing header and information to the second module that is leaf node searcher. The second module compares header and rules in the leaf node until best matching rule get or no matching rule condition occurs. The leaf node searcher consists of two comparator block works in parallel that allows two rules to be searched on each memory access.

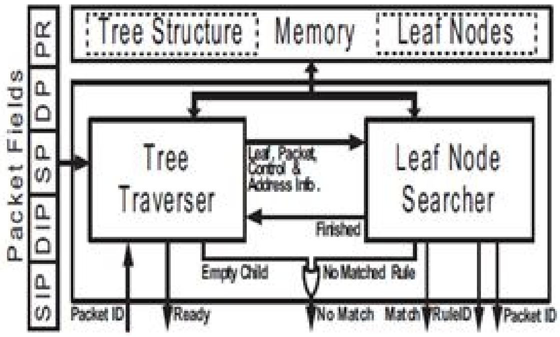


Figure. 4: Architecture of packet classification engine.

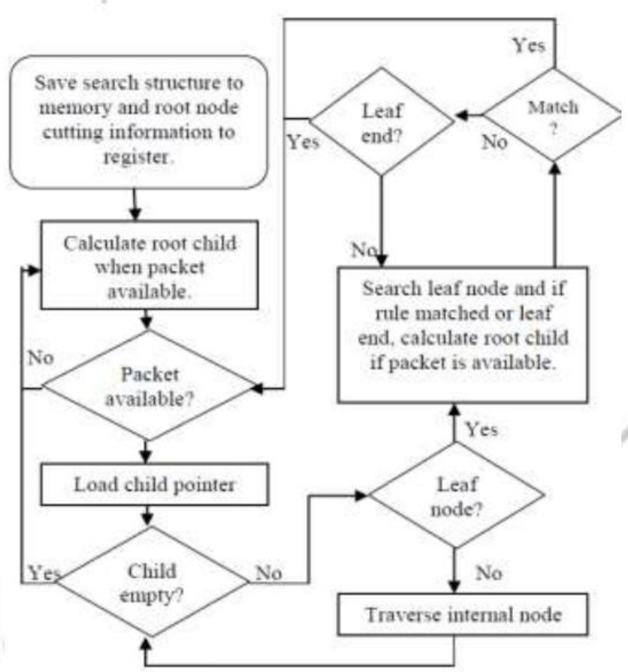

Save search structure to memory and root node cutting information to register.
Calculate root child when packet available.
Packet available?
No
Load child pointer
Child empty?
Yes
No
Leaf node?
Yes
No
Traverse internal node
Search leaf node and if rule matched or leaf end, calculate root child if packet is available.
Match ?
Yes
No
Leaf end?
Yes
No

Figure.5: Packet classification engine Operation.

In tree traverser, information on root node of decision tree is stored in registers so that tree traverser is classify new packets and old packets compared with rules in leaf nodes. Also pipelining increases throughput having single packet with two clock cycles if root and leaf node of decision tree doesn’t having more than two rules.

Figure 5 shows the Flowchart explain the detail operation of packet classification. The engine designed in such a way that it traverses root node or internal node in one memory access. Also it can search leaf node with two rules on every memory access.

The classifier having multiple packet classification engines works in parallel. The maximum speed of engine is much slower than speed of the internal memory of FPGAs because of the comparator blocks used in engine that generates logic delays. So that multiple engines needed and classifier maximizes throughput. Another reason of using multiple engines is that rule sets having many wildcard entries that are divided into groups. After dividing the rule sets it is easier to build decision tree with small leaf nodes which ultimately increases throughput and reduces memory.

Figure 6 shows the architecture of classifier containing total eight packet classification engine working in parallel. Classifier takes advantage of FPGA which having dual internal memory so that two classifiers working in parallel and uses same memory. Each classifier reads data from data port and packet header buffers stores the header fields of incoming packet. It works on the basis of first come first served basis and produces packet ID output that with matching rules are outputted in same order of incoming packet. The four packet classification engines of classifier working in parallel at the same clock speed but out of phase. Sorter logic block are used for matching rule IDs are outputted in order of packet input. Also sorter block accept match, No Match, Rule ID, Packet ID signals and according to the highest priority rule it produces many results.

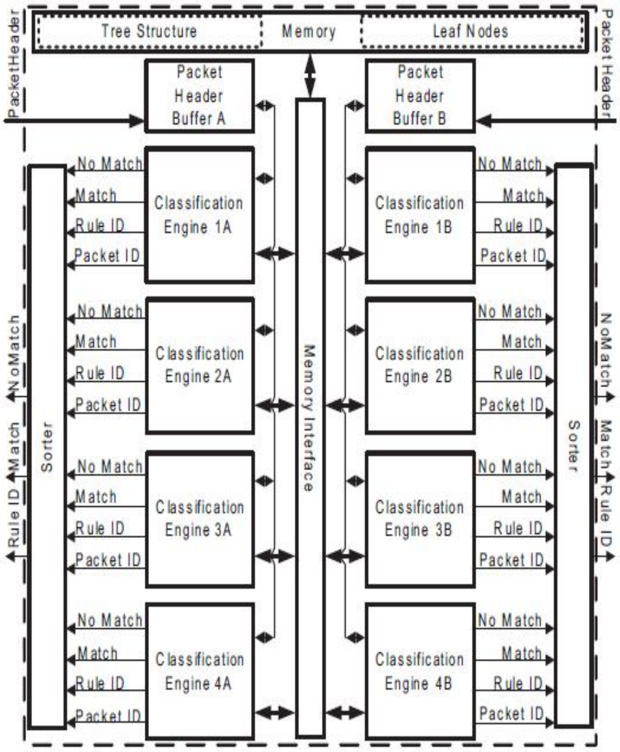


Figure 6: Architecture of the classifier.

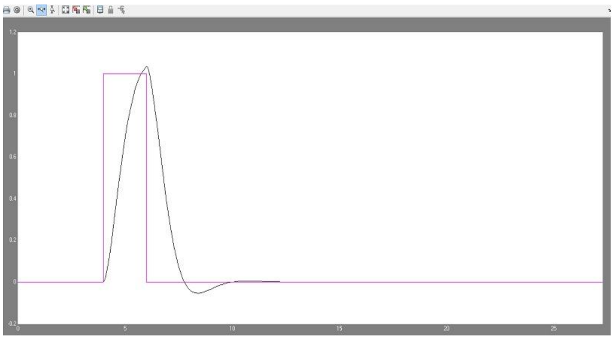
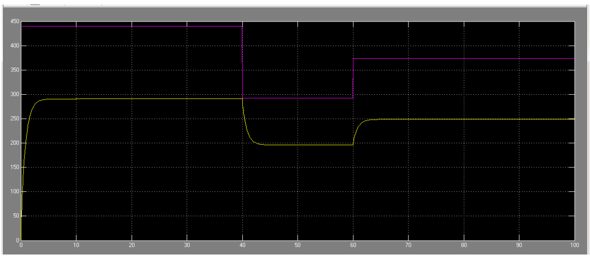


Figure 3. Servo Response of Trial and Error Method

The above figure shows the servo response of Trial and Error Method, the process output meet the desired set point quickly. The controller parameters are
Kp=8; Ki=3; Kd=1



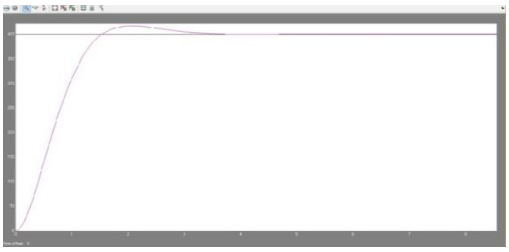
Time (sec)

Figure 4. Servo response comparison of Ziegler-Nichols PID-controller and fuzzy PID-Controller

The above figure shows the servo response of Ziegler-Nichols PID Controller and Fuzzy PID Controller. In the Fuzzy PID controller, The process output meet thedesired set point quickly with less oscillation than the Ziegler NicholsPIDcontroller.The controller parameters are:

Kp=0.7878;Ki=3.0857;Kd=0.7714

Fig 5 shows schematic block diagram of Model Reference Adaptive Controller



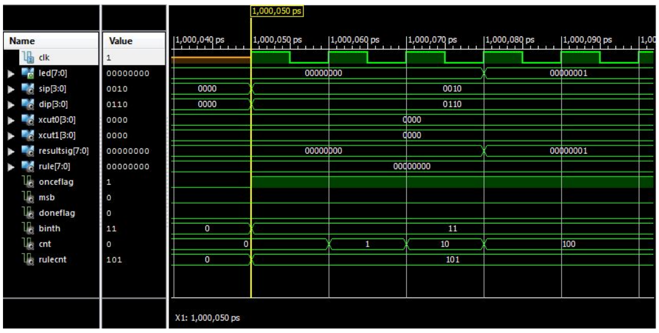
Time (sec)

Figure 5. MordenAdapative reference control

The above figure shows the output response of MARC. It meets the desired set point quickly without overshoot. The output responses clearly show Model Reference Adaptive controller is the best controller it reach the set point quickly without overshoot then all other controllers.

V. RESULTS

The system is implemented on Xilinx Vivado 2016.1 for both decision tree based algorithms HiCuts and HyperCuts algorithms. The different results are obtained by varying rule sets from 5 rules to 25 rules. Same design is implemented on Hardware namely Genesys-2 Kintex-7 FPGA board and observed matching rules results on led.



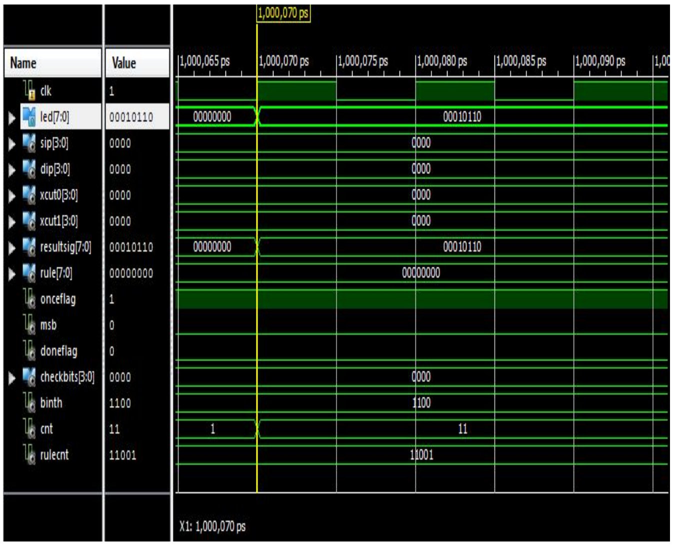
Name	Value
clk	1
led[7:0]	00000000
sip[3:0]	0010
dip[3:0]	0110
xcut0[3:0]	0000
xcut1[3:0]	0000
resultsig[7:0]	00000000
rule[7:0]	00000000
onceflag	1
msb	0
doneflag	0
binth	11
cnt	0
rulecnt	101

X1: 1,000,050 ps

Figure 7: Simulation result for HiCuts and HyperCuts algorithms for 5 rules.

The simulation shown in figure 7 indicates that at the rising edge of the clock we get output and that store in output named as led. The rulecnt indicates that total number of rules that can be used to simulate the both the algorithms. Here rulecnt shows that 5 rules are used. Also by giving input values sip and dip in the program then it can generates output. In this particular simulation sip and dip values are matching with the rule 1 so led output indicates value as “00000001”. In particular 5 rules example binth values taken as 3 for both HiCuts and HyperCuts algorithms but depth of decision tree is more that is 2 in HiCuts while that of HyperCuts is1.

Similarly we have done the simulation for 10 rules, 15 rules and 20 rules and obtain similar results. Finally we have done the simulation for the total 25 numbers of rules with 2 fields sip and dip. The figure indicates that rulecnt value is 11001 and matches with the rule 22 as indicates in output led value “00010110”. The depth of decision tree and binth values is 1 and 12 simultaneously for both HiCuts and HyperCuts algorithm.



Name	Value
clk	1
led[7:0]	00010110
sip[3:0]	0000
dip[3:0]	0000
xcut0[3:0]	0000
xcut1[3:0]	0000
resultsig[7:0]	00010110
rule[7:0]	00000000
onceflag	1
msb	0
doneflag	0
checkbits[3:0]	0000
binth	1100
cnt	11
rulecnt	11001

Figure 8: Simulation result for HiCuts and HyperCuts algorithms for 5 rules.

Thus it is clear that, as the values of sip and dip is changes the particular algorithms is work for any number of rules and finding out matching rules and store the results in output led.

As we have to increases the rules values from 5 to 25 by keeping difference of 5, the depth of decision tree is varies in both the algorithms and binth values are increases. Also both these algorithms have different execution time as well as memory requirements is also varies that are given in summary tables and graph which indicates both these parameters along with depth of decision tree and binth values.

Table I: Summary of Hicuts and HyperCuts simulation results (B: binth value, D: Max. tree depth, Execution time in ns and memory in kilo bytes)

No. of rules	HiCuts Algorithm				HyperCuts Algorithm			
	B	D	Execution time (ns)	Memory (KB)	B	D	Execution time(ns)	Memory (KB)
5	3	2	0.8324	428719	3	1	0.779	432802
10	6	2	0.9388	442056.8	7	1	0.779	440904.8
15	7	2	0.8614	443490.4	10	1	0.779	424328
20	10	1	0.7944	441045.6	10	1	0.779	426492
25	12	1	0.7884	430447.2	12	1	0.779	434261.6

The summary table and graph of these algorithms for execution time with respect to number of rules is indicated that execution time for HyperCuts is less as that of HiCuts algorithm for any number of rule count. Thus HyperCuts is faster than HiCuts algorithm. Also depth of decision tree is always 1 for HyperCuts that is less than HiCuts algorithm. So that in many cases memory requirements for HyperCuts is less than that of HiCuts algorithm. Thus we can say that in all ways HyperCuts decision tree algorithm is superior than HiCuts algorithm.

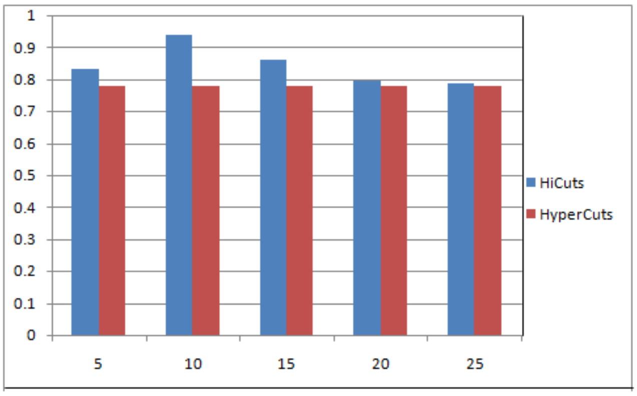
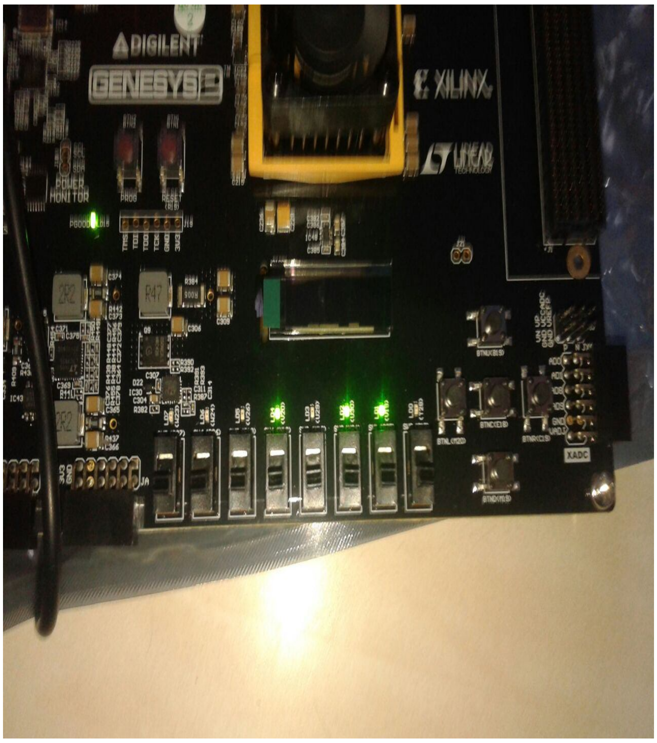


Figure 9: Execution time for HiCuts and HyprCuts with respect to number of rules.

Hardware Implementations Results:

Finally we obtain same results on Kintex-7 FPGA board as shown below:



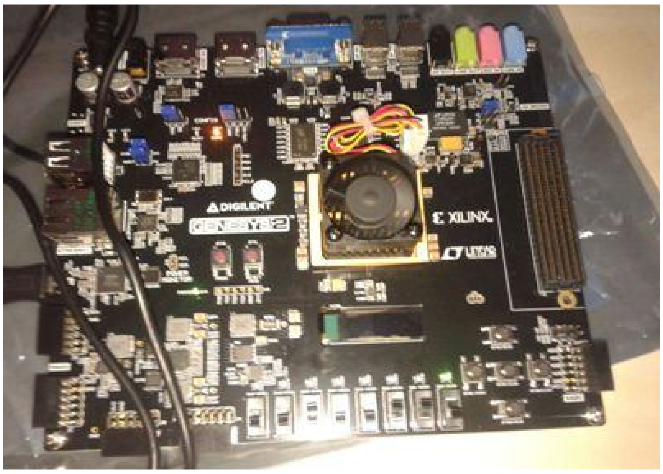


Figure 10: Genesys-2 Board Results

In the figure 10 LEDs are blinking which shows the results of matching rule number 16.

VI. CONCLUSION

This Paper explains the basics of packet classifications algorithms. Basically different algorithms are used to classify the packets but we considered two decision tree based packet classification algorithms namely HiCuts and HyperCuts. These algorithms explained in detail, simulated on latest software Xilinx Vivado2016.1 and implemented on high performance Genesys 2 Kintex-7 development board.

The objectives of Paper are to developed hardware system which can classify the packets using decision tree based algorithms. The system was able to classify packets, increases the performance with the less memory requirements than other packet classification algorithms.

The HiCuts and HyperCuts both algorithms are simulated on software and implemented on kit. Thus it is observed that HyperCuts algorithm is superior than HiCuts with respect to many parameters such as depth of decision tree, execution time and memory requirements with the different number of rules. The depth of decision tree is 1 and execution time for Hypercuts algorithms is 0.779ns for all cases which is less than that of HiCuts algorithms. Also in many cases the memory requirements of HyperCuts is less than HiCuts algorithm. Thus both the algorithms work better way on hardware.

REFERENCES

[1] D. E. Taylor, “Survey and Taxonomy of Packet Classification Techniques,” ACM computing Surveys, Vol. 37, No. 3, Sept. 2005, pp.238-275.

[2] P. Gupta and N. Mckeown, “Classification using Hierachical Intelligent Cuttings,” IEEE Micro, Vol. 20, No.1, Feb 2000, pp.34-41.

[3] S.Singh, F. Baboescu, G.Varghese and J. Wang, “Packet Classification using Multidimensional Cutting,” in Proc. ACM special Interest Group Data Commun Conf., Aug. 2003, pp. 213-224.

[4] Alan Kennedy and Xiaojun Wang, “Ultra High Throughput Low Power Packet Classification,” IEEE Transaction on Very large scale integration (VLSI) systems, Vol.22, No.2, Feb. 2014, pp.286-299.

[5] Weirong Jiang and Vector K. Prasanna, “Scalable Packet Classification on FPGA,” IEEE Transactions on very large scale integration (VLSI) systems, Vol. 20, No. 9, Sept. 2012, pp. 1668-1680.

[6] Pankanj Gupta and Nick Mckeown, “Packet classification on multiple fields,” in Proc. ACM Special Interest Group Data Commun Conf.,
Sept. 1999, pp.147-160.

[7] Haoyu Song and John W. Lockwood, “Efficient Packet Classification for Network Intrusion Detection using FPGA”, ACM1, Feb.2005.

[8] R. Sathesh Raaj and J. Kumarnath, “FPGA Based Packet Classification using Multi-pipeline Architecture”, IJCSMC, Vol.4, Issue.4 , Apr. 2015, pp. 541-548.