International Journal of Combined Research & Development (IJCRD)
eISSN:2321-225X; pISSN:2321-2241 Volume: 2; Issue: 6; June -2014

FPGA Implementation of Binary-tree-based High Speed Packet Classification System

Bhakti Sidrai Tumari[1], Lakshmipriya W[2]
[1] Student, Dept.of ECE(VLSI & ES), Banglore, bhaktitumari@gmail.com
[2] Assistant Professor, Dept.of ECE, Banglore, lakshmipriya@tjohngroup.com

Abstract: Packet classification involving multiple fields is used in the area of network intrusion detection, as well as to provide quality of service and value-added network services. With the ever increasing growth of the Internet and packet transfer rates, the number of rules needed to be handled simultaneously in support of these services has also increased. Field-Programmable Gate Arrays (FPGAs) provide good platforms for hardware-software co-designs that can yield high processing efficiency for highly complex applications. In the network intrusion detection system (NIDS), there is a limitation on the speed of software-based packet classification because of the processor performance, the serial program execution and so on. It has become a great challenge to develop scalable solutions for next-generation packet classification that support higher throughput, larger rule sets and more packet header fields. For low-cost high performance embedded networking applications, the best solution could be doing packet classification by special designed hardware, which can effectively release the burden of system CPU. The binary tree structure is generated through pre-processing on computer, which does not influence the searching speed of FPGA. During the packet header division, the division field is dynamic and selected according to the rules.

KEYWORDS – PACKET CLASSIFICATION, FPGA, BINARY TREE, NIDS

I.NTRODUCTION

It is invariably the ultimate goal to improve the efficiency and security of network operation in today's Internet world. Access control, traffic engineering, intrusion detection, and many other network services require the discrimination of packets based on the multiple fields of packet headers, which is called packet classification.

One of the fundamental challenges in designing high speed router is packet classification [1]. The router has to support firewall processing, quality of service differentiation, policy routing, and other value added services which is enabled by packet classification. When a packet arrives at a router, its header is compared with a set of rules. Each rule can have one or more fields and their associated value, and an action to be taken if matched. A packet is considered matching a rule only if it matches all the fields within that rule. Until recently, Internet routers provided only "best-effort" service, servicing packets in a first-come-first-served manner. Routers are now called upon to provide different qualities of service to different applications which means routers need new mechanisms such as admission control, resource reservation, per-flow queuing, and fair scheduling. All of these mechanisms require the router to distinguish packets belonging to different flows. Flows are specified by rules applied to incoming packets. Collection of rules is called as a classifier. Each rule specifies a flow that a packet may belong to based on some criteria applied to the packet header.

The identifying of packets for quality of service (QoS) packets can be classified by source and destination ports and address and protocol type. The process of categorizing packets into "flows" in an Internet router is called packet classification. All packets belonging to the same flow obey a pre-defined rule and are processed in a similar manner by the router. For example, all packets with the same source and destination IP addresses may be defined to form a flow. Packet classification is needed for non "best effort" services, such as firewalls and quality of service; services that require the capability to distinguish and isolate traffic in different flows for suitable processing. In general, packet classification on multiple fields is a difficult problem.

On multi-dimension packet classification there has been a lot of research going on over the past decades. Most existing solutions cannot meet the performance requirement. On the one hand, software solutions based on multi-core network processors for high performance packet classification have good flexibility and programmability, but they inherently lack high parallelism and abundant on-chip memory [2][3].

On the other hand, hardware solutions are mainly based on TCAM (Ternary Contend Addressable Memory) and Bloom Filter. TCAM-based solutions can reach high speed performance while they sacrifice scalability, programmability and power efficiency. TCAM-based solutions also suffer from a range-to-prefix conversion problem, making it difficult to support large and complex rule sets. Although Bloom-Filter-based algorithm can process prefix-type rules well, it is not good at range-type rules. What's more, it induces complex conflict [4] [5] [6].

II.THEORY OF BTPCF

An IP packet is usually classified based on the five fields in the packet header: the 32-bit source/destination IP addresses (denoted SA/DA), 16-bit source/destination port numbers (denoted SP/DP), and 8-bit transport layer protocol. In our system, the 32-bit source/destination IP addresses is used to classify the packet.

The binary decision-tree data structure is built by carefully preprocessing. Every father node has two child nodes, and every node has two packet header fields: the source and destination IP addresses (the address is usually not an exact number, but an address range). The steps for preprocessing are as follows:

1) At first, generate the father node, which contains the full header fields' ranges: source and destination IP addresses' ranges are both 0.0.0.0""255.255.255.255;
2) Generate two child nodes connected to the father node which also have two packet header fields; ,
3) Choose one of father node's packet header fields' ranges to divide equally into two small rule ranges (0.0.0.0""127.255.255.255,128.0.0.0""255.255.255.) which are connected to the two child nodes respectively; another packet header fields' range does not change. Choosing which header field to divide depends on which one has less Snort rules;
4) At last, determining the threshold (marked as Leaf_max) by simulation on PC according to the particular rule sets. If the child node contains more rule numbers than the threshold, divide the child node again until the rule number the new child nodes contain is not bigger than the threshold. If all rule numbers that the child nodes contain are not bigger than the threshold, the division is stopped, which means the binary decision-tree data structure has been built successfully.

When a packet arrives, the BTPCF algorithm traverses the decision tree to find the leaf node, which stores the rule that contains the header ranges of the input packet. If the leaf node can be found, it means this packet header is contained in Snort; otherwise, it means not. The input packet only needs to travel along one branch to find the leaf node, and then compare with rules the leaf node contains to find the rule that contains the header ranges. Preprocessing software AddrSearch, which uses binary tree algorithm to generate balance binary decision-tree, preprocess the Snort rule sets and sends them to the FPGA through Ethernet.

Table 1 is mapped to a 2- dimension space (in figure 1). At first, set the threshold to be 2 Gust for example).Then generate the root node which contains the full header fields' ranges: source and destination IP addresses' range are both 0.0.0.0--255.255.255.255. From the figure 1 we can see that the left field still contains 5 rules (RO,R1,R2,R3 and R4), so it needs to be divided further. The right field contains 3 rules (RO,R4,R5), so it needs to be divided, too. Field 1 and field 2 are kept dividing until the rule numbers the leaf nodes contain are not bigger than 2. After dividing 3 times, a 3-level-depth tree structure is generated (in figure 1),which contains 5 middle nodes and 6 leaf nodes. The rule numbers each leaf node contains are not bigger than 2, which means the binary-decision-tree data structure has been built completely. Each father node contains the information of division field, division portion and child nodes' addresses.

We use IP packet (source address: 96.78.0.51, destination address: 45.34.23.50) as an example to show the search procedure. This IP packet goes along dashed line in figure 2 to traverse the branch until reaching the leaf node 2_2. Compared with the rules R2 and R4 contained by the leaf node 2_2, it matches R4.

As every IP packet matches the root node, every IP packet search from the root node. If it reaches the leaf node, the search is stopped. Otherwise, judging which child node the IP packet belongs to, continuing new search from his child node.

Division boundary can be within a rule's field range, which leads to that two or more leaf nodes contain this rule; it takes more memory space to store, as well as increases the tree's depth. The threshold (Leaf_max) is very important to the BTPCF, which determines the tree's depth, memory space and liner searching speed.

Table 1 An example rule set used in binary tree algorithm

Rule Fieldl(source IP address)	Field 2(destination IP address)
R0 90.75.0.0 ""156.75.255.255	202.38.0.0 "" 202.38.255.255
Rl 45.78.64.0 "" 45.78.64.255	202.38.0.0 "" 202.38.0.255
R2 96.78.0.0 "" 96.78.255.255	96.45.72.0 "" 96.45.72.255
R3 62.38.64.128""62.38.64.255	96.45.72.0 "" 150.45.72.255
R4 96.78.0.50 "" 215.34.23.10	45.34.23.44 "" 80.34.0.0
R5 180.25.33.0"" 200.120.33.0	192.168.10.0 "" 200.30.34.5

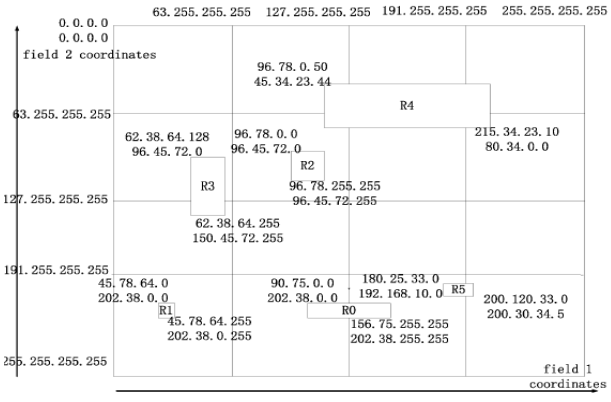

Fig. 1 Motion of binary tree for rules field division

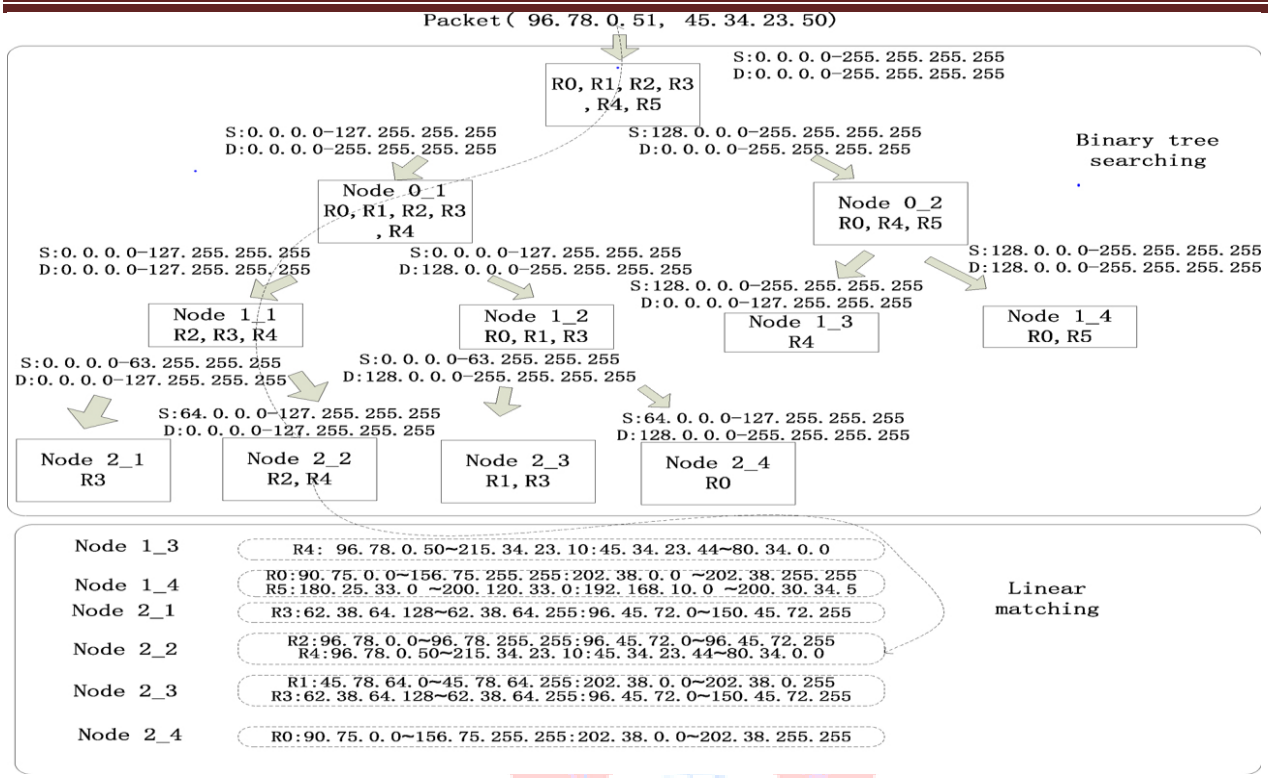

Fig. 2 Structure of binary tree and linear searching for rules

III.HARDWARE DESIGN OF BTPCF

A Hardware design principle ofBTPCF

Binary tree structure is saved in "tree node memory". Each node (include the middle nodes and leaf nodes) uses 32 bit memory space; all child nodes of the same father nodeare stored in a continuous memory space and the child nodes from the same level are stored in a continuous memory space in the order from left to right. Every node's 32 bit data provides the information that can traverse all his child nodes. The field ranges of rules contained by the leaf nodes are stored in the "upper boundary value memory" and "lower boundary value memory" (the "upper boundary value memory" stores the upper boundary value of the field range and the "lower boundary value memory" stores the lower boundary value). The address of upper boundary value stored in "upper boundary value memory" is corresponding to the same rule's lower boundary value's address in "lower boundary value memory". The leaf node's 32 bit data contains the rule's base address in "upper boundary value memory" and "lower boundary value memory". The binary tree node's storage structure is shown in figure 2.

Figure 3 describes nodes' data structure. In binary tree, every node's 32 bit data is set as follows: if the node is a middle node, setting the flag bit (F, bit 0) to 0, otherwise, 1; division field (bit 3-1) stands for this node's division field number (field 1 or field 2); division position (pos, bit 8-4) is used to index the child node which contains the input IP packet, then uses this index to traverse the binary tree's branch; child node's base address (bit 31-17) stands for base address of this node's child node in the memory space. For example, node 0_1 is a middle node, whose flag bit is 0, division field is 1, division position is 31

and child node's base address is Ox0003. If the node is leafnode, then flag bit is 1. Rule number (NoR, bit 3-1) stands for how many rules this leaf node contains. Rule's base address (Addr, bit 23-4) stands for this leaf node's base address in "upper boundary value memory" and "lower boundary value memory". For example, node 2_1 is leaf node, whose flag is 1, rule number NoR=1 and Rule's base address Addr= OxOOOOO.

The description above describes the binary tree algorithm's preprocessing and packet classification engine's hardware design principle under the premise that rule and packet only have two 32 bit fields. However, it is also suitable for multiple fields.

B Binary tree searching process

During the tree searching process, the system should often access the "tree searching memory" , which stores the structure of binary tree. The on-chip-RAM is used as "tree searching memory" to supply higher access speed. The tree searching controller controls the "tree searching memory" to execute the searching process, starting from the root node, traversing the tree as shown in figure 2 until finding the leaf node. The searching process is shown in figure 4:

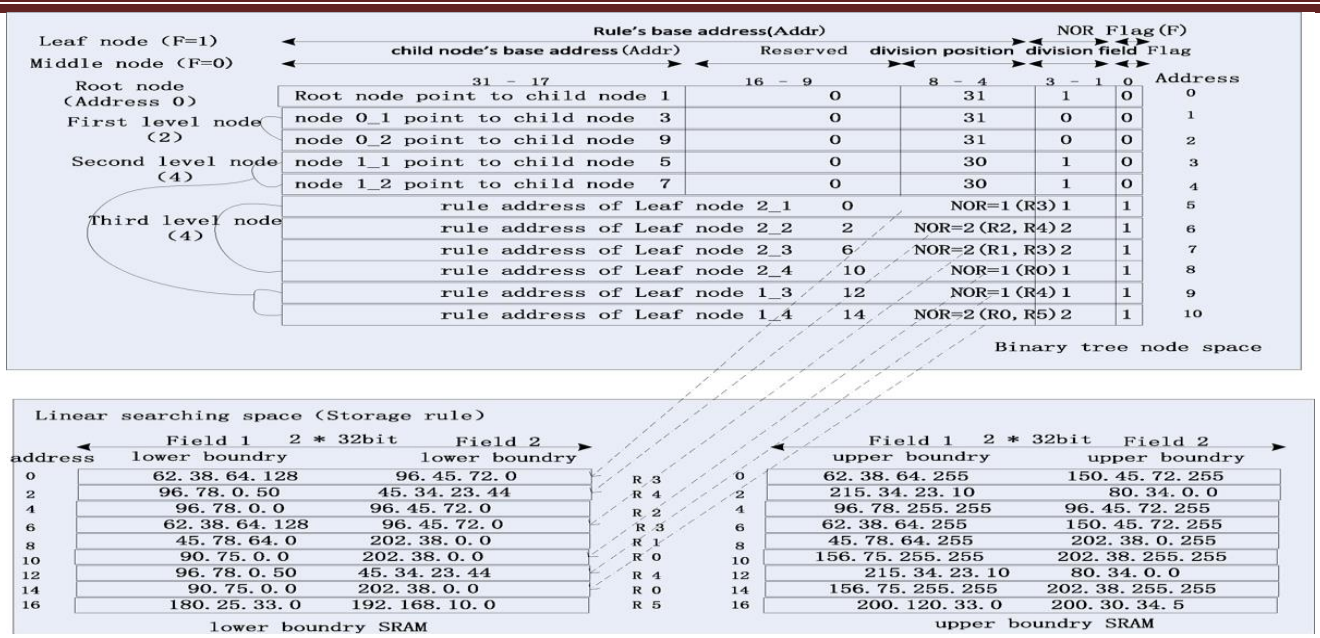

Fig. 3 storage space of Binary tree nodes and structure of linear searching

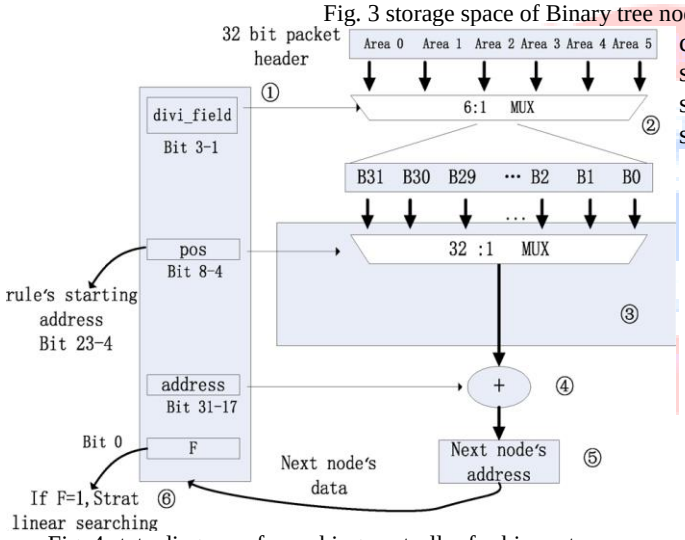

Fig. 4 state diagram of searching controller for binary tree

1) Extract the IP packet's field according to the division field's data;
2) Transmit the extracted data to the buffer (B31-BO);
3) Use the division position (pos) to calculate the next offset address;
4) Add the offset address and child node's base address together to get the next child node's address in the tree searching memory;
5) Read next child node's 32 bit data from the tree searching memory;
6) Judge whether this node is middle node (F=O). If F=O, it means this node is middle node, continue the search; otherwise, it means not, stop the search.

From figure 4, we can see that each tree search is in 6 steps. The steps (5) and (6), accessing the on-chip-RAM, consume 1 clock cycle. The steps (1) to (4), which are realized by combinational logic, consume 1 clock cycle. After the tree search processing, tree searching controller sends the tree search results, the data of the found leaf node, to the linear search engine's buffer

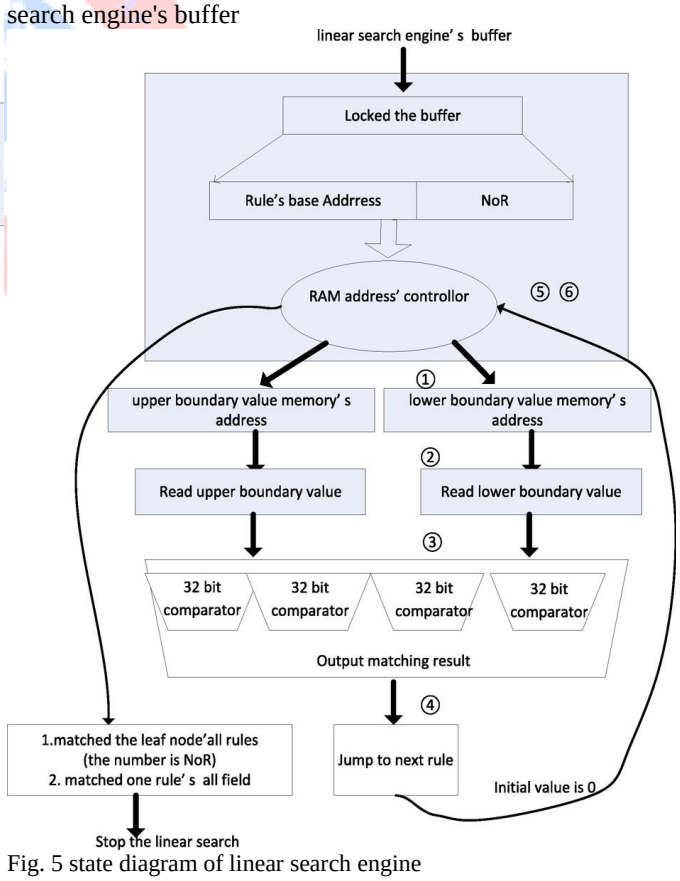

Fig. 5 state diagram of linear search engine

Figure 5 shows the linear search engine's processing steps. At first, the tree search results are read from the linear search engine's buffer, and the linear tree search is started as follows:

1) Generate rule's base address according to the address ofupper and lower boundary value memory;
2) Read data from the upper boundary value memory and lower boundary value memory;
3) Compare the data of IP packet's field with the data of rule's field;
4) If they match, stop the linear tree search; otherwise, go to step (5).
5) & (6) use the result of step (4) and other information, such as Rule number (NoR), to calculate the addresses of next accessing rule's upper and lower boundary value memory.

In the linear search described above, step (1) and (2) consume 2 clock cycles to access the memory; (3) to (6) consume 1 clock cycle; so the total consumed time is 3 clock cycles. During the linear search, 4 comparators are used to realize parallel processing. If all rules of the found out leaf node (the number is NOR) have been compared, the linear tree search is stopped.

IV.EXPERIMENTALRESULTS

RTL VIEW

After the HDL synthesis phase of the synthesis process, a schematic representation of synthesized source file can be displayed. This schematic shows a representation of the pre-optimized design in terms of generic symbols, such as adders, multipliers, counters, AND gates, and OR gates. Viewing this schematic may help you discover design issues early in the design process.

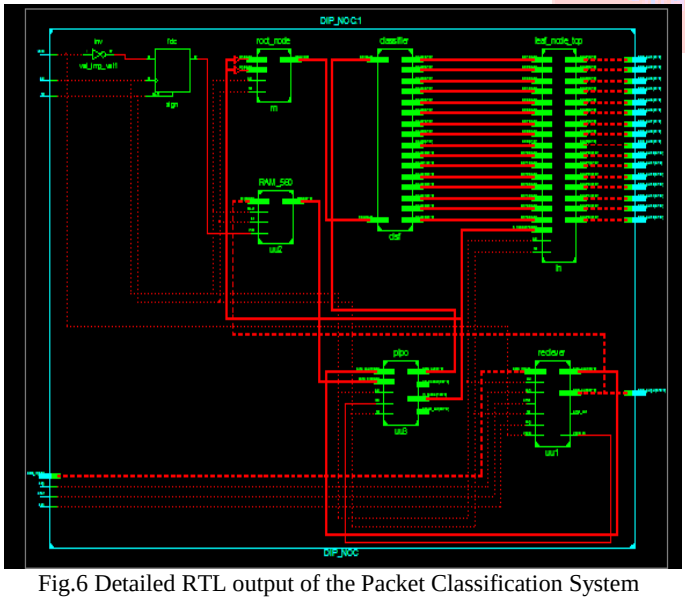

Fig.6 Detailed RTL output of the Packet Classification System

Simulation result

Figure 7 shows simulated result of Packet Classification when start of packet becomes high the input packets will be transmitted to the specified destination address generated. The memory location will keep on increasing as many packets are arriving.

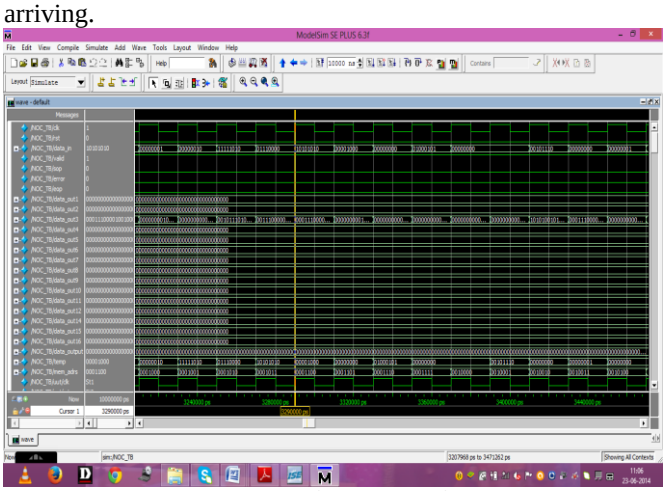

Fig.7 Modelsim Simulated Output of Packet Classification System

Output Using Chip Scope Pro Analyzer

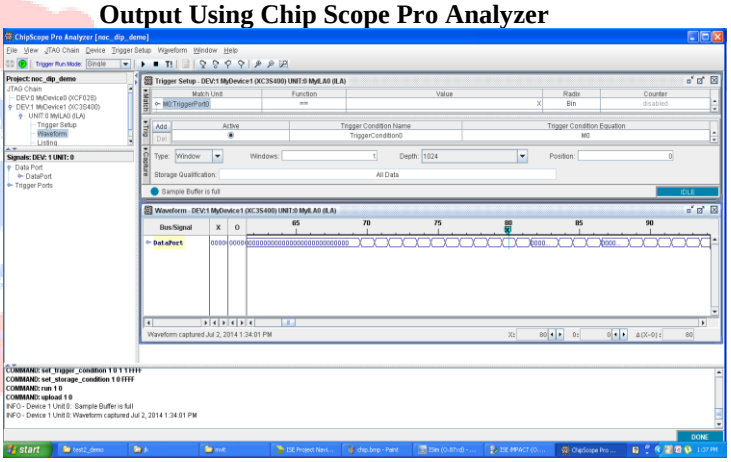

Fig.8 Hardware Controlled Output Using Chip Scope Pro Analyzer

As shown in figure 8 when the reset is made it as high chip scope analyzer gives the output as zero when reset is made low analyzer will display the 128 bits output. The output is controlled by the hardware.

V.CONCLUSION

Binary Tree based packet classification system is a memory efficient and searching quick data structure for packet classification based on FPGA is done. Using the binary tree structure enhances the speed of preprocessing and packet classification based on FPGA.

The rule's preprocessing time only includes the time from the generating of random rule sets to the completion of building binary tree structure, does not include the time of traversing binary tree, reading the node's data and linear search. In binary tree algorithm, the preprocessing time is

about 0.000563s (rule number is 5000) and about 0.050331s (rule number is 10,000). In HiCuts algorithm, the preprocessing time is about 8 s (rule number is 5000), about 33 s (rule number is 10,000) and about 140 s (rule number is 20,000). Therefore, the binary tree's preprocessing time is much smaller than HiCuts algorithm's.

The packet classification consists of three parts: packet extraction (extract the fields' value from the input IP packet), binary tree search and linear search. As these three parts processed parallel, and the later uses the front parts' results, we should make these three parts' processing time equivalent to enhance the system's processing ability. The packet extraction's processing time is fixed; binary tree search's processing time depends on binary tree's depth (Tree_depth) and linear search's processing time depends on the number of leaf nodes' rules(NoR). In addition, NoR is relevant to the threshold (Leaf_max).

Taking advantage of pipeline and parallel processing improves the system's processing ability greatly. Results show that binary tree algorithm consumes less memory space than HiCuts algorithm (when the rule number is 10,000, it only consumes 200 KB memory space); when the rule number is 10,000, preprocessing time for building tree structure is less than 0.051s, which means it takes less time than HiCuts algorithm(33 s).

This system can be used in a variety of network security systems with some other modification.

REFERENCES

[1] Yeim-Kuan Chang, Han-Chen Chen, Layered Cutting Scheme for Packet Classification, Advanced Information Networking and Applications. 2011 IEEE International Conference, pp. 675 - 681.

[2] Yuhua Chen, Oladapo oguntoyinb0 , Power efficient packet classification using cascaded bloom filter and off-the-shelf ternary CAM for WDM networks. Computer Communications, Volume 32, Issue 2, pp. 349-356, February 2009.

[3] Erdem, 0.; Hoang Le; Prasanna, V:K.Clustered Hierarchical Search Structure for Large-Scale Packet Classification on FPGA. Source: 2011 International Conference on Field Programmable Logic and Applications, pp. 201-206, 2011.

[4] Qi, Yaxuan; Fong, Jeffrey; Jiang, Weirong; Xu, Bo; Li, Jun; Prasanna, Viktor. Multi-dimensional packet classification on FPGA: 100 Gbps and beyond. Source: Proceedings – 2010 International Conference on Field-Programmable Technology, FPT'10, pp 241-248, 2010.

[5] Derek Pao, Yiu Keung Li, Peng Zhou, Efficient packet classification using TCAMs. Computer Networks, Volume 50, Issue 18, pp. 3523-3535, December 2006.

[6] A.G Alagu Priya, Hyesook Lim, Hierarchical packet classification using a Bloom filter and rule-priority tries. Computer Communications, Volume33, Issue 10, pp. 1215-1226, June 2010.

[7] Gupta, Pankaj, McKeown, Nick. Classifying packets with hierarchical intelligent cuttings, IEEE Micro, Volume 20, Issue 1, pp. 34-41,2000.

[8] Jiang W., Prasanna v. K.,Scalable Packet Classification on FPGA,Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, Volume:20, Issue: 9 ,pp. 1668-80,2012.

[9] Wang Yong-gang, Zhang Tao, Zheng Yu-feng, Yang Yang, Realization of FPGA-based packet classification in embedded system, Instrumentation and Measurement Technology Conference, 2009. I2MTC '09. IEEE, pp. 938 - 942, 2009.

[10] Gupta P., McKeown N., Classifying packets with hierarchical intelligent cuttings, IEEE Micro, Volume:20, pp. 34- 41, 2000

Author Biographies

Bhakti Sidrai Tumari received diploma and engineering degree in electronics and communication engineering from Board of Technical Education Banglore and Visvesvaraya Technological University,Belgaum Karnataka India , in 2009 and 2012. She is currently working for MTech degree in VLSI and Embedded Systems at T John Institute of Technology Banglore.

Lakshmipriya Wudali as assistant professor T John Institute of Technology Banglore