Set Pruning Segment Trees for Packet Classification

Yeim-Kuan Chang and Hsin-Mao Chen
Department of Computer Science and Information Engineering,
National Cheng Kung University,
Tainan, Taiwan
{ykchang, P76984500}@mail.ncku.edu.tw

***Abstract*—Nowadays, multi-field packet classification is one of the most important technologies to support various services in next generation routers. In this paper, we propose a segment tree based parallel SRAM-based pipelined architecture called Set Pruning Segment Trees (SPST) for multi-dimensional packet classification. For solving the memory blowup problem, a grouping scheme called Partition by Length (PL) is used to reduce the rule duplications in SPST. Additionally, we also propose an optimization called Set Pruning Multi-way Segment Trees (SPMST) to reduce the tree level and hardware cost. The key feature of our proposed architecture is that memory consumption is reduced significantly regardless of the characteristics of various rule tables. The proposed pipelined architecture can achieve a throughput of 89.4 Gbps for minimum sized packets with dual port memory on Xilinx Virtex-5 FPGA device.**

Keywords-segment tree; elementary interval; pipeline; FPGA; packet classification

I. INTRODUCTION

The workload of next generation routers increases as the various services on the Internet grow rapidly. These Internet services include virtual private network (VPN), quality of service (QoS), network security, and firewall. Packet classification is the core functionality in the high-performance routers [17] for supporting these Internet services. It is always a challenge to develop a high-speed packet classification algorithm in routers to sustain the ever-growing Internet traffic required by these Internet services. To match a rule, packet classification needs to compare multiple header values of each incoming packet with the field values of all the rules in the rule table. In a common rule table, there are five fields that are source and destination IP address fields in the form of prefixes of variable length, source and destination port fields in the form of ranges of consecutive numbers, and protocol field in the form of singleton value.

The packet classification approaches can be classified into software and hardware based schemes. The software-based solutions include decision schemes such as Hierarchical tries [14], Grid of tries [14], Set Pruning tries [14], HiCuts [6], and HyperCuts [15]. Dynamic Grid of Segment Trees (DGST) [4] is proposed to allow dynamic insertions and deletions of ranges. Hierarchical tries is a multi-dimensional binary tries. The disadvantage of Hierarchical tries is that search operations can not be completed without backtracks. Set pruning tries solves the backtracking problem by duplicating rules. Unfortunately, duplicating rules cause the memory blowup problem. Grid of tries is another way of solving the backtracking problem for 2-dimensional classification. The key feature is to use pre-computations to set switch pointers pointing to the part of data structure containing the matched rules. It is effective in dealing with prefixes, but it is not suitable for other fields such as port ranges. To make Grid of Trie work for rules containing range fields, all the range field values must be converted to prefixes first. However, Prefixes are limited ranges and the size is a power of two. In the worst case, a W-bit range needs to be converted to 2W – 2 prefixes. Thus, converting ranges to prefixes becomes another source of suplications. HiCuts, HyperCuts and DGST are suitable for range fields, but those memory usages are too large to implement on Field Programmable Gate Array (FPGA) easily.

The hardware-based solutions include static random access memory (SRAM) architecture such as Improved HyperCuts [9], Set Pruning Multi-Bit Trie (SPMT) [3], and Power Saved HyperCuts [11]. The hash-based schemes are Dual Stage Bloom Filter Classification (2sBFCE) [12], Bloom Based Packet Classification (B2PC) [13], and Nest Level Tuple Merging and Cross-product (NTLMC) [5], etc. There are schemes which are based on ternary content addressable memories (TCAM) such as BV-TCAM [16]. Most TCAM-based schemes are like the binary trie based schemes which are not suitable for range fields. Moreover, the power consumption is a pending issue. The memory usage of hash-based schemes is efficient, but the throughput is limited due to resolve false positives problems. It is important to pay attention to Improved HyperCuts and SPMT which have higher throughputs and both of them are trie-based pipelined architectures. Unfortunately, those schemes have used all the SRAMs for implementing large rule table such as ACL1_10K on FPGA. They could not support all the large rule tables. On the other hand, there are some problems on traditional trie-based pipelined architecture. The size of the memory in each stage is unbalanced and the utilization rate of SRAMs in some pipeline stages is inefficient. For the above problem, some IP Lookup trie-based pipelined architectures have been proposed for memory balancing in [8] and [10].

In order to solve the problem caused by range fields and achieve a high throughput, we propose a packet classification scheme called Set Pruning Segment Trees (SPST) in this paper. SPST is very suitable for the pipelined architecture which is a hierarchical scheme based on segment tree by replacing the binary trie in set pruning tries with segment trees. We also use the rule partitioning scheme, Partition by Length (PL), which divide rules into subgroups by prefix lengths to reduce the rule duplications in SPST. Each SPST is built for each subgroup and all the SPSTs are searched in parallel to obtain the best

1550-445X/11 $26.00 © 2011 IEEE
DOI 10.1109/AINA.2011.69

TABLE I. A SAMPLE RULE TABLE

Rule	SA	DA	SP	DP	PORT	PRIORITY	ACTION
R1	0*	10*	[0:31]	[0:31]	TCP	1	Accept
R2	00*	11*	[0:15]	[21:21]	TCP	2	Accept
R3	011*	00*	[16:23]	[20:22]	TCP	3	Accept
R4	10*	1*	[31:31]	[0:15]	TCP	4	Accept
R5	*	00*	[16:31]	[5:8]	TCP	5	Accept
R6	0*	01*	[15:16]	[10:15]	UDP	6	Deny
R7	00*	10*	[0:7]	[16:31]	TCP	7	Deny
R8	0*	*	[0:31]	[0:31]	UDP	8	Deny

TABLE II. MINUS – 1 ENDPOINTS OF SA AND DA IN TABLE I

Rule	SA			DA		
	Prefix	Start	Finsh	Prefix	Start	Finsh
R1	0*	-	15	10*	15	23
R2	00*	-	7	11*	23	31
R3	011*	11	15	00*	-	7
R4	10*	15	23	1*	15	31
R5	*	0	31	00*	-	7
R6	0*	0	15	01*	7	15
R7	00*	0	7	10*	15	23
R8	0*	0	15	*	-	31

matched rule. In order to further reduce the hardware cost in our pipelined architecture, we will also propose a Set Pruning Multiway Segment Trees to merge the stages using small amount memory.

The rest of the paper is organized as follows. In section 2, the background of packet classification is introduced. In section 3, we review the related work on software and hardware based schemes. The section 4 describes our proposed scheme. Section 5 describes the implementation of our proposed design. Section 6 is our optimization. Section 7 presents the results of the comparisons. And finally, the Section 8 is the conclusion.

II. BACKGROUND

In the general packet classification problem, query packets are classified by searching the rule table to determine an action (e.g., acceptance or denial) to be applied on the packets. Each rule R contains a number of fields as well as the priority and the associated action. Traditionally, the rule contains five fields: the source and destination address which are variable length prefixes, source and destination port which are range numbers, and protocol which is explicit value. When a packet P matches a rule R, it means that the all header fields of rule R matches all corresponding fields of the incoming packet P. It is possible that a packet may match multiple rules. And one of the packet classification problems is to determine the highest priority rule which called best match rule. Table I is a sample 5-field rule table. If the field values of the incoming packet (SA, DA, SP, DP, Protocol) = (00000, 10000, 21, 21, UDP), the matched rules from Table I are R1 and R8. The priority of R1 is higher than R8. So the best match rule is R1.

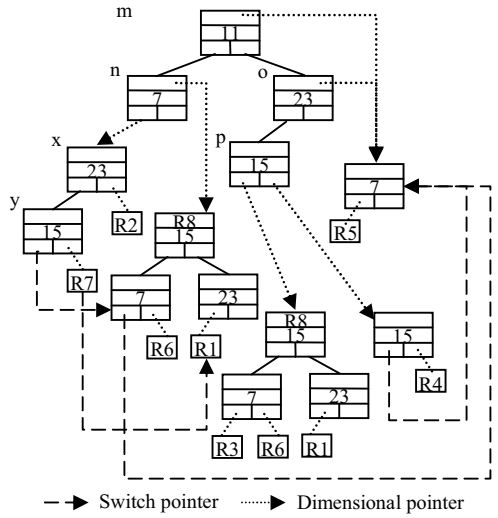

Figure 1. A possible 2D DGST without extended pointer

III. RELATED WORK

In this section we review the software based scheme called Dynamic Grid of Segment Tree. On the other hand, we also review the hardware based scheme called Set Pruning Multi-Bit Trie and the important issues of trie-based pipelined architecture.

A. Dynamic Grid of Segment Tree

Dynamic Segment Tree (DST) [2] is proposed to solve the IP lookup problem caused by range fields and allow dynamic insertions and deletions of ranges. DST is built from the distinct endpoints of ranges which obtains by minus-1 endpoint scheme. Table II is the minus-1 endpoint of SA and DA in Table I. The interval between two endpoints is called elementary intervals (EIs) [1]. Each node in DST covers a number of consecutive EIs. For examples, each leaf nodes represents an EI, and the parent node of leaf nodes represents the EIs of the leaf nodes. DST improves the traditional Segment Tree by supporting dynamic routing tables.

Dynamic Grid of Segment Tree (DGST) [4] that is similar to Grid of Trie replaces the binary tries in Grid of Trie by DSTs to inherit advantage of reducing memory usage and avoiding backtracking. There are switch pointers and extended pointers to speed up the search operations by finding potential match rules in DGST. The nodes also contain dimension pointers to pointer to next dimensional DST. In DST, each node distinguishes three intervals which are left interval, right interval, and union of right and left intervals (called canonical set C) by the key of the node. Figure 1 is a 2-dimension example of DGST without extended pointer built according to Table II. To find the best match rule, the trace of an incoming packet with (SA, DA) = (6, 16) is *m*-*n*-*x*-*y*-*R*7-*R*1. The matched rules are R1 and R7, and the best matched rule is R1. Because

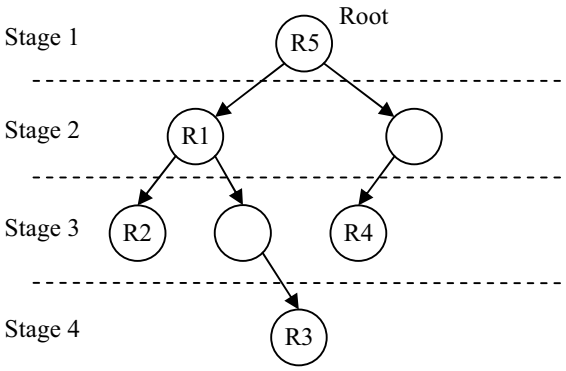

Figure 2. A traditional binary trie-based pipeline architecture according to SA of Table I.

of the feature of Grid of Tries, DGST extended to 3 or more dimensions will also need backtracking.

B. Set Pruning Multi-Bit Trie

In [3], the authors proposed a trie-based SRAM pipelined architecture for packet classification. In order to solve the memory blowup problem caused by rule duplications, the authors proposed two rule grouping schemes called *Partition by Wildcards* (*PW*) and *Partition by Length* (*PL*). PW divides rule table into 16 subgroups. Those are classified according to the wildcards values in SA, DA, SP, and DP fields (excluding protocol field). When most of prefixes in the rule table are short (say length = 1 or 2), most subgroups will be empty and the rule duplication is still high. Thus, scheme PL deals with the rule duplication of short prefixes. PL divides rule table into k subgroups by select $k - 1$ different prefix lengths in the first field. The prefix lengths are selected by computing the Duplicating Cost (DC) of the rule table. The set of k prefix lengths is selected if it produces the smallest DC. Because the traditional multi-bit trie is suitable for the IP Lookup pipelined architecture [7], the authors developed a multi-bit scheme called Set Pruning Multi-Bit Trie (SPMT) to share nodes for reducing the number of pipeline stages and avoiding the redundant data structures.

C. Balance The Memory

As shown in Figure 2, each trie level is assigned to a different pipeline stage in traditional trie-based pipelined architecture. However, there is only one node in first stage and the stage containing the maximum number of nodes depends on the rule table. The hardware clock rate is determined by the largest stage and it's called the unbalanced memory problem. In [10], the authors proposed two different schemes to incrementally update and balance memory usage for IP Lookup trie-based pipelined architecture. The authors use the No Operation Performed (NOP) instructions to make the nodes in the same trie level map onto different pipeline stage for balancing memory. For dealing with the route updates, the traditional pipelined engines need re-map the entire trie onto the pipeline. The update cost is high and the authors proposed two trade-off schemes to reduce the cost of updating the pipeline based on only mapping the newly inserted nodes onto the pipeline. Additionally, the authors also proposed another pipelined architecture that assigns few stages to be the external stages for supporting large rule tables.

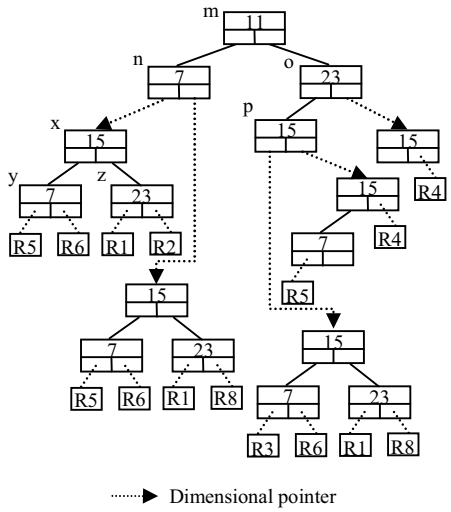

Figure 3. A possible 2D SPST according to Table I

IV. Proposed Scheme

In [4], the authors solve the range problem in multi-dimensional packet classification. There is a problem that the DGST's data structure is too complex to implement in FPGA effectively. Another problem is that DGST will cause backtrack problem in D-dimension DGST (D>2) and thus the traditional trie-based pipelined architecture is not suitable. Another scheme called SPMT is a very fast pipelined architecture and solves the memory blowup problem for being implemented on FPGA. Because the ranges need to covert into a number of prefixes and the trie level may be the maximum. Therefore, there are a large number of pipeline stages and usage of memory. In this section, we propose a new scheme in the subsequent two subsections to solve the above problems.

1) *Set Pruning Segment Trees (SPST): SPST is a segment tree based data structure which is suitable for solving 5-dimensional packet classification problem. SPST is also the Set Pruning trie based data structure which can be implemented in FPGA easily.*
2) *Grouping Set Pruning Segment Tree: In SPST, there is also rule duplicating problem. We use the Partition by Length to reduce the memory usage. But we do not reduce the rule duplication as much as possible. There is a trade-off between memories consumption and hardware cost.*

A. Set Pruning Segment Trees (SPST)

In SPST, each node contains the key (endpoint of range based on minus-1 endpoint scheme), two link pointers (pointer to left and right children), and the link type (indicating that the pointer children are the same dimension or next dimension). Constructing SPST is similar to the Set Pruning Trie construction. In the first step, the first-dimensional segment tree is constructed. The endpoints are used to create the

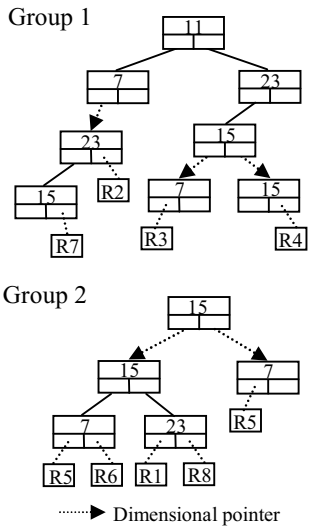

Figure 4. A possible PL 2D SPST according to Table I

elementary intervals. Then, the Segment Tree is constructed by using the most balancing Segment Tree to minimize the tree level and duplicating all the rules to the leaf nodes. After constructing the first dimensional Segment Tree, we repeat the above steps and will obtain the 5-dimensional SPST. In the last dimensional Segment Tree, the leaf nodes only store the highest priority rule of the best match rule. Figure 2 shows an example of 2-dimensional SPST built according to source and destination addresses in Table I. The search in SPST is from root to leaf node. Each node compares the key and the corresponding fields of the query packet. If the key is small than or equal to the corresponding field value of the packet, it travels to the left child or vice versa. Repeating the above step to the leaf node will find the best match rule. For the example in Figure 3, if the header values of the incoming packet is (SA, DA) = (6, 16), the trace is m-n-x-z-R1. And the best match rule is R1.

B. Grouping Set Pruning Segment Tree

Because all the rules are duplicated into the leaf nodes, the search process is simpler. But as shown in Figure 3, the memory blowup problem is in existence. In order to reduce the duplicated rules, the partition scheme from [3] is employed. We compare the efficiency of Partition by Wildcards (PW) and Partition by length (PL) in our scheme. Table II shows the node numbers of SPST divided by PW and PL. Those three different type tables are generated by ClassBench [18]. By PW, the SPST is divided into 16 subgroups some of which are empty. In Table III, we can notice that we just divide rule table into two subgroups by PL, but the efficiency results are better than PW in each table. As a result, we determine to use the PL grouping scheme to improve our SPST. Figure 4 is an example of 2D PL SPST built according to Table 1. If the header values of the incoming packet is also (6, 16), the match rule of Group 1 is R7 and Group 2 is R1. And the best match rule is R1.

After choosing the partition scheme, we consider how many subgroups are appropriate. In general, we will minimize the Duplication Cost (DC) as much as possible. DC is defined as a

TABLE III. COMPARE THE PW AND PL GROUPING SCHEMES.

Rule Table	Partition by Wildcards		Partition by Length	
	# of nodes	# of groups	# of nodes	# of groups
ACL1_1K	9297	5	7766	2
ACL1_5K	55034	5	36566	2
ACL1_10K	226805	5	193876	2
IPC1_1K	63528	12	26173	2
IPC1_5K	962500	12	171699	2
IPC1_10K	57515415	12	225036	2
FW2_1K	40706	6	15136	2
FW2_5K	806723	6	80505	2
FW2_10K	2503799	6	161733	2

TABLE IV. DUPLICATION COST BY PW

# of groups	Duplication Cost		
	ACL1	IPC1	FW2
2	27576	28422	13501
3	17265	27445	12401
4	17170		12188

value of additional cost due to duplicate rule when constructing a SPST. We calculate the sum of all the rules copy times when constructing a SPST. If the value of DC is small, it means that we don't need waste a lot of additional cost. But in the next section we will propose a parallel architecture. It means the more groups we divide, the more hardware cost is needed. On the other hand, before the rule table is divided up into the numbers of the lowest DC groups, the effect of reducing rule duplication decreases quickly. As shown in Table IV, we calculate the DC from three rule tables containing 10k rules. In ACL1 table, the DC using 3 groups is about 10 thousands less than that of using 2 groups. But the difference of DC between using 3 groups and 4 groups is less than one hundred. This situation is similar in IPC1 table. For the above reasons, we determine to trade off between the hardware cost and the memory usage. If the DC difference between two groups is not over the threshold, we will choose a smaller k to divide the rule table into k subgroups in SPST. In addition, For FW2 table, the DC of 3 groups is the minimum, so we don't show the DC that uses 4 groups.

V. HARDWARER IMPLEMENTATION

In this section, the proposed hardware design is given. We design a pipelined and parallel architecture to improve the throughput of the SPST groups. About the pipelined architecture, each node in SPST is marked as a level according the tree level in SPST. Then the nodes with the same level are mapped to the same pipeline stage. The nodes in the different dimension Segment Tree are not mapped to the same pipeline stage. Each query packet searched from the root to the leaf nodes is the same as traveling the pipeline architecture stage by stage. Figure 5 is our designed pipelined architecture. The actions of each pipeline stage are described as follows:

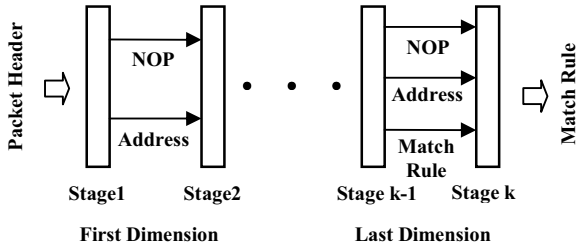

Figure 5. Pipeline Architecure

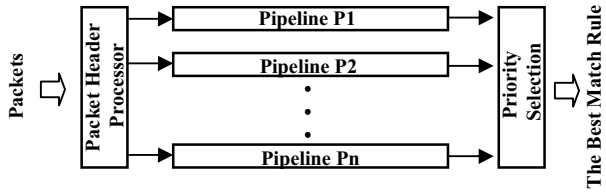

Figure 6. Parallel Architecure

1) *The memory of each piepline stage is accessed to obtain the key of the node in this stage, the next address, and the next stage NOP signal.*
2) *If the import NOP signal is asserted, this stage passes the import address to the next stage. The NOP singal is deasserted until the next dimensional stage.*
3) *If the import NOP signal is deasserted, this stage will compare the key and the corresponding field of the query packet to determine the address of the next stage.*
4) *In the last dimensional pipeline stage, if the NOP signal is asserted, this stage will obtain the information of match rule and pass it to the last stage.*

As shown in Figure 6, we also take advantage of the FPGA with massive parallelism to propose a parallel architecture. Additionally, FPGA also provides a dual-port Block RAMs. It contributes to improve the throughput massively. We construct a set of pipelines each of which is built from different SPST subgroup. When a query packet is coming, the packet header will be imported to all the pipeline sets. Each pipeline will execute independently and output the result to the Rule Selection module individually. Then the Rule Selection module will output the highest priority rule. This rule is the final result.

VI. OPTIMIZATION

Memory balancing is a major problem in trie based pipeline architecture [10]. In our pipelined architecture, the SRAM in each pipeline is unbalance. And the clock rate of the hardware is determined by the pipeline with the largest SRAM. For the above reason, we propose an optimization scheme called Set Pruning Multiway Segment Trees (SPMST) to combine the stages with few nodes for reducing the trie level and hardware cost in our pipelined architecture. We combine the stages in the first dimension of SPMST and the combined SRAMs would not be larger than the largest SRAM in SPST. In SPMST, the node and the children nodes are combined to form a multiway node. The data structure of SPMST is similar to SPST. Each

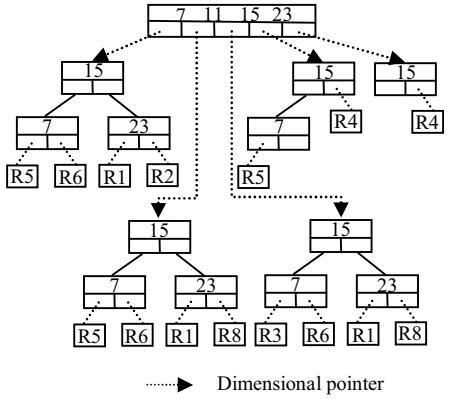

Figure 7. A possible 2D SPMST according to Table II

node contains the key set, the corresponding link pointers and link type. Figure 7 is a possible 2D SPMST which is built from Table II. Compared with Figure 3, the tree level of first dimension decreases.

VII. PERFORMANCE

In this section, we present the performance results of our proposed scheme and other schemes on memory, hardware cost and throughput. We implement our scheme by using 4-way Segment Tree in the first dimensional Segment Tree. The node size of original node is 54 bits which needs 3 Block RAMs and the 4-way node is 108 bits which needs 6 Block RAMs. The implementation result of pipeline stages in first dimension decreases to half of the pipeline stages needed in the original architecture.

The memory performance of our proposed scheme and without partition is shown in Table V. The rule tables are obtained from ClassBench [18] and three real-life rule tables. Compared with our scheme using no partitioning, it shows that partition scheme is effective on the data structure we proposed. The memory usage is reduced effectively. Based on our results, our scheme can save a lot of memory regardless of the characteristics of the rule table. The memory usage of each rule table is small enough to fit in the Block RAM of FPGA. Furthermore, the highest level in ACL_10K is 19, in FW2_10K is 17, and in IPC1_10K is 24. It means that our proposed pipeline stages are equal to or less than the traditional IP lookup trie-based pipelined architecture. Additionally, ACL1_10K and FW2_10K were partitioned into three groups. And IPC1_10K was partitioned into four groups. It shows that we do not need a lot of parallel pipeline search engines. For the above two reasons, we can conclude that our scheme also can reduce a large number of hardware cost. We will show the detailed results below with other schemes.

Table VI is the FPGA implementation results. We use Xilinx ISE 10.1 development tools to implement our proposed scheme on FPGA. In Table VI, there are other methods' results such as Improved HyperCuts proposed in [9] and Set Pruning Multi-bit Trie Partition by Wildcards and Length combined (SPMT PW and PL) proposed in [3]. For having a fair comparison with Improved HyperCuts and SPMT with PW and

TABLE V. MEMORY PERFORMANCE OF SPMST

Rule table	# of group	The level	# of node	Total Memory (Kb)	No partition Memory (Mb)
ACL1_1K	2	17	6085	255.65	0.49
ACL1_5K	3	19	27681	960.9102	2.39
ACL1_10K	3	19	95013	3618	21.07
IPC1_1K	2	14	12593	398.60	50.06
IPC1_5K	3	16	66448	2287.943	1746.36
IPC1_10K	3	17	145464	5017.68	2230.03
FW2_1K	3	19	18977	597.16	65.24
FW2_5K	3	19	91813	2941.251	1020.55
FW2_10K	4	24	141813	5005.68	Overflow

TABLE VI. HARDWARE RESOURCE COMPARSION

	Improved HyperCuts	SPMT by PW and PL	SPMST
# of slices / utilization	10307 / 33%	6854 / 24%	2136 / 6%
# of Block RAMs / utilization	407 / 89%	429 / 94%	129 / 28%
Frequency (MHz)	125.4	173.02	139.76
Throughput (Gbps)	80.23	110.73	89.4

TABLE VII. COMPARING THROUGHPUT WITH OTHEER METHODS

Approaches	# of rules	Throughput (Gbps)
SPMT PW and PL	9603	110.73
SPMT PW	4451	107.16
MSPST	9603	89.4
Improved HyperCuts	9603	80.23
B2PC in ASIC	3300	13.60
NTLMC	12507	12.16
Power Saved HyperCuts	≒25000	10.24
BV-TCAM	222	10.00
2sBFCE	4000	5.86

PL, we also use the same FPGA device, Xilinx Virtex-5 XCVFX200T [19] with '-2' speed grade, and dual-port memory. The three methods' implementation results are obtained by using ACL1_10k rule table. Compared with Improved HyperCuts, we can see that our hardware cost is lower. The slice utilization is less than twenty percent of Improved HyperCuts (33% vs. 6%). The Block RAMs utilization is less than one third of Improved HyperCuts (89% vs. 28%). And the throughput is also higher than Improved HyperCuts (80.23 Gbps vs. 89.4 Gbps). On the other hand, our proposed scheme is compared with the SPMT using PW and PL partitioning schemes. Although our throughput is lower (110.73 Gbps vs. 89.4 Gbps), our hardware cost is lower. Our slice utilization is equal to twenty-five percent of SPMT PW and PL (24% vs. 6%). And the Block RAMs utilization is also less than one third of SPMT PW and PL (94% vs. 28%). Also, according to the above results, these three schemes can achieve the throughput of OC-768 but hardware cost and memory usage of our proposed scheme are the lowest. As we can see, Xilinx Virtex-5 XC5VFX200T is sufficient to support the three methods with 10k rules. But Improved HyperCuts and SPMT PW and PL have used almost Block RAMs for implementing ACL1_10K rule table. The XC5VFX200T is not sufficient to support both of them for FW2_10K or IPC1_10K rule tables. The memory usage and the Block RAMs of our proposed scheme are both small regardless of the type of rule table. Our proposed scheme is accurately implemented with the XC5VFX200T.

Table VII compares the throughput of our scheme and other schemes. Our proposed scheme is implemented with Xilinx Virtex-5 XC5VFX200T and packet size is assumed to be 40 bytes. We can see that our throughput is better than all the schemes except for SPMT PW and PL. But as showd in Table V, our slice and Block RAMs utilization are less than SPMT PW and PL.

VIII. CONCLUSION

In this paper, we proposed a SRAM-based pipelined architecture for packet classification and reducing the memory usage and hardware cost for being implemented on FPGA with high throughput. First, we proposed a segment tree based data structure called Set Pruning Segment Trees which is suitable for range fields and mapping onto pipelined architecture. Because of the Set Pruning property, we need to solve the memory blowup problem caused by rule duplication. We partition the rule table into some subgroups by Partition by Length grouping scheme to reduce the memory usage and trade-off the hardware cost. Finally, we proposed an improved data structure called Set Pruning Multiway Segment Trees to combine the nodes in the stages using less memory for reducing the tree level and hardware cost. We implemented our proposed scheme with Xilinx Virtex-5 FPGA. Based on our performance experiments, our scheme can achieve 89.4 Gbps with dual port memory and support all kind of large rule tables.

REFERENCES

[1] M.D. Breg, M.V. Kreveld, M. Overmars, and O. Schwarzkopf, Computational Geometry: Algorithms and Applications. Springer Verlag, 1997.

[2] Yeim-Kuan Chang and Yung-Chieh Lin, "Dynamic Segment Trees for Ranges and Prefixes", IEEE Transations on Computers, vol. 56, no. 6, pages. 769-784, June 2007.

[3] Yeim-Kuan Chang, Yi-Shang Lin, and Cheng-Chien Su, "A High-Speed and Memory Efficient Pipeline Architecture for Packet Classification", Proc. IEEE FCCM, pages. 215-218, 2010.

[4] Yeim-Kuan Chang, Y.-C. Lin, and C.-Y. Lin, "Grid of Segment Trees for Packet Classification", In IEEE AINA, pages. 1144-1149, 2010.

[5] S. Dharmappurikar, H. Song, J. Turner, and J. Lockwood, "Fast Packet Classification Using Bloom Filters", In ACM/IEEE ANCS, 2006.

[6] P. Gupta and N. McKeown, "Classifying packets with hierarchical intelligent cuttings", IEEE Micro, vol. 20(1), pages. 34-41, 2001.

[7] W. Jiang and V. K. Prasanna, "A Memory-Balanced Linear Pipeline Architecture for Trie-based IP Lookup", In IEEE HOTI, pages, 83-90, 2007.

[8] W. Jiang, Q. Wang, and V. K. Prasanna, "Beyond TCAMs: An SRAM-based parallel multi-pipeline architecture for terabit IP lookup", in Proc. INFOCOM, pages. 1786-1794, 2008.

[9] W. Jiang and V. K. Prasanna, "Large-Scale Wire-Speed Packet Classification on FPGAs", In ACM/SIGDA FPGA, 2009.

[10] W. Jiang and V. K. Prasanna, "Towards Practical Architectures for SRAM-based Pipelined Lookup Engines", in Proc. INFOCOM'10 Work-in-Progress track, Mar. 2010.

[11] A. Kennedy, X. Wang, Z. Liu, and B. Liu, "Low Power Architecture for High Speed Packet Classification", In ACM/IEEE ANCS, 2008.

[12] A. Nikitakis and I. Papaefstathiou, "A Memory-Efficient FPGA-Based Classification Engine", In IEEE FCCM, 2008.

[13] I. Papaefstathiou and V. Papaefstathiou, "Memory-Efficient 5D Packet Classification at 40 Gbps", In IEEE FCCM, 2008.

[14] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, "Fast and Scable Layer Four Switching", ACM SIGCOMM Computer Communication Review, vol. 28, pages. 191-202, October 1998.

[15] S. Singh, F. Baboescu, G. Varghese, and J. Wang, "Packet classification using multidimensional cutting", In Proc. SIGCOMM, pages. 213-224, 2003.

[16] H. Song and J. W. Lockwood, "Efficient Packet Classification for Network Intrusion Detection Using FPGA", in Proc. FPGA, pages. 238-245, 2005.

[17] D. E. Taylor, "Survey and Taxonomy of Packet Classification Techniques", ACM Computing Surveys, vol. 37, no. 3, pages. 238-275, Sep. 2005.

[18] D. E. Taylor and J. S. Turner, "ClassBench: A Packet Classification Benchmark", IEEE/ACM Transation on Network, vol. 15, no. 3, pages. 499-511, June 2007.

[19] Xilinx, "Virtex-5 Family Overiew", Product Specification, DS100 (v5.0), Feb. 6, 2009, at http://www.xilinx.com.