ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING

National Conference on Advanced Innovation in Engineering and Technology (NCAIET-2015)

Alva's Institute of Engineering and Technology, Moodbidri

Vol. 3,Special Issue 1,April 2015

Modular Architecture for High Speed Packet Classification using SBV Algorithm

Sushanth Anil Lobo[1], Roshan S Shetty[2]

PG Scholar, Department of Electronics and Communication Engineering, Alva's Institute of Engineering and Technology, Moodbidri, Karnataka, India[1]

Assistant Professor, Department of Electronics and Communication Engineering, Alva's Institute of Engineering and Technology, Moodbidri, Karnataka, India[2]

Abstract: In network infrastructure packet classification is a core function for various applications which is widely used. Performing wire speed classification is becoming a challenge because of demand in increasing throughput. Performance of packet classification these days depends on rulesets and its characteristics. A high speed packet classification based on Bit-Vector (BV) based architecture implemented on FPGA (Field Programmable Gate Array) is proposed. StrideBV is the algorithm introduced and BV architecture is modularized to achieve better scalability than BV traditional methods. The solution introduced here is ruleset-feature independent, the performance is mostly guaranteed for any ruleset regardless the ruleset and its composition. The proposed design is implemented in SPARTAN3 or higher devices FPGA by using Xilinx ISE 13.4 and simulated in modelsim 6.3f. The Chipscope pro Analyzer is used to view the execution results of FPGA.

Keywords: Packet classification, firewall, router, network security, FPGA, networking.

I. INTRODUCTION

Packet classification is an important technique used for various purposes in networking equipment. Application of packet classification is diverse including access control lists, network security, traffic accounting. Performing packet classification is done by checking one or more header fields of an incoming packet against a set of rules which are predefined; they are usually termed as a classifier or a ruleset. Set of rules or filters is called classifier. For example every rule in firewall could specify a set of destination and source addresses and may have permissions such as 'deny' or 'allow' action with itself. Rules could be based on many fields of the packet including 2, 3, 4 layer and can be 5 addressing including protocol information. Initial filter to network where network traffic classification is done into flows based on certain predefined ruleset is packet classification. Multiple fields of packet header is inspected and compared with best effort IP forwarding wherein only the IP address of destination is inspected. It is more challenging specially in environments wherein wire-speed checking of packets is mandatory [8]. Performing packet classification faster is challenging since it involved inspection of multiple fields against a ruleset possibly which contains thousands of rules. Performing such operation at wire speed is challenging with the increasing in throughput demands in modern networks [9]. Many hardware platforms have been used for packet classification in previous methods. Field Programmable Gate Arrays (FPGAs) offer re-configurability and deliver good performance due to the custom built nature of the architecture. Application Specific Integrated Circuits (ASICs) on the offer little flexibility in the architecture once fabricated, but gives superior performance compared with Field Programmable Gate Arrays. These platforms are widely used in many networking applications to implement methods such as multi-gigabit packet classification forwarding engines.

The output of hardware based routers can be dramatically increased by employing pipelining techniques. Packet classification performs searching the table of filters to assign a flow identifier for the highest priority filter that matches the packet in all fields. The returning flow ID indicates the action that is next applied to the packet. An example of a packet filter set is shown in Table 1. The standard five-tuple fields are shown in separate columns for the purpose of clarity. This table illustrates a small sample packet filter set with a very few bits for simplicity. Filters are usually sorted in the order of priority in the filter set. In this table, x indicates wildcards inserted in any location of the fields. When a packet arrives, the first (which is generally the best) matching filter in the set is to be found in packet classifiers. The matching filter should match the filter in all five fields. The address (index) of the matching filter is used to point to the action that needs to be applied to the packet. Most packet classifiers store the index to indicate the action that is going to be processed on the packet afterward. For example, if a packet consisting source address of 0101, destination address of 0011, source port of 4, destination port of 6, and protocol field of TCP is received, the packet classifier should report the second filter as the matching filter and, hence, would forward the packet to output port 5. It is obvious that arriving packets may result in multiple filters

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING

National Conference on Advanced Innovation in Engineering and Technology (NCAIET-2015)

Alva's Institute of Engineering and Technology, Moodbidri

Vol. 3,Special Issue 1,April 2015

matching the packet in the set. In this example, the arriving packet matches the seventh filter in addition to the second one. The general packet classification process only reports one filter (which is the highest priority filter) in case of multiple matches. However, we elaborate why some networking applications require finding multiple matches in the packet filter set. Filter fields are combination of prefixes, wildcards, and exact values [9].

Table 1: Packet Filter Set

Filter	Src. Addr	Dest Addr	Src. Port	Dest. Port	Prot	Action
1	1001	01xx	2-4	7	TCP	Forward 3
2	01xx	00xx	3-9	2-6	TCP	Forward 5
3	110x	10xx	1-7	4-6	UDP	Accept
4	1101	101x	x	X	ICMP	Queue
5	00xx	010x	4	5	UDP	Accept
6	111x	01xx	x	X	X	Drop
7	0101	xxxx	x	X	X	Forward 4
8	1xxx	0xxx	x	X	X	Drop

The rest of this paper is organized as follows. Section II includes the related work which provides an overview of different approaches used for packet classification. Section III describes the packet classification processes and the proposed work. Section IV includes the Hardware Architecture. Section V includes results of part of present work. Conclusion and future work is included in Section VI.

II. LITERATURE SURVEY

Many Packet classification algorithms are extensively studied in the past. A survey can be found in [7]. Many of the said algorithms can be divided into two categories: decision-tree-based and decomposition-based approaches. Here, only decomposition based approaches is discussed as the Stride BV belongs to this category. T.V. Lakshman and D. Stiliadis in their research work based on Decomposition-based algorithms (e.g. Parallel Bit - Vector (BV) [2]), perform independent search on every field and finally combine all the results from all fields. Those algorithms are desirable for hardware implementation because of their parallel search on many multiple fields. However, substantial storage is usually required to merge the independent searched results to obtain the end result. Considering the BV algorithm as a example, it does the parallel lookup on every individual field. The lookup on every field returns a bit-vector where every bit represents a rule. A bit is high if the corresponding rule is matched on that field; a bit is low if the corresponding rule is not matched on that field. The result of bitwise AND operation on those bit-vectors indicate the set of rules which matches a given packet. The BV algorithm provides a high throughput at the low cost at low memory efficiency.

By combining BV and TCAMs algorithm, Song et al. [3] presented a architecture called BV-TCAM for multi match packet classification. A TCAM performed prefix or exact match, while a multi-bit trie implemented in Tree Bitmap [1] is used for source or destination port lookup. The authors never reported the actual FPGA implementation results, though they claimed that the whole circuit for 222 rules consumed less than 10 percent of the available logic and less than 20 percent of the available Block RAMs of a Xilinx XCV2000E FPGA. They also predicted the design after pipelining which can achieve 10 Gbps throughput when implement on advanced FPGAs.

D. Taylor and J. Turner [5] introduced Distributed Cross-producting of Field Label (DCFL), which is also decomposition based algorithm leveraging many observations of the structure of real filter. They decomposed the multi-field searching problem and used independent search engines, which operated in parallel to find many matching conditions for every filter field. Instead of considering bit-vectors, DCFL uses a network of useful aggregation nodes, which employed Bloom Filters and encoded intermediate search results. Which resulted, the algorithm avoided the exponential increase in time or space which incurred when performing the operation in single step.

In [6] an algorithm namely Field Split Bit-Vector (FSBV) was introduced which performs individual field search like a chain of sub-field search, in form of bit-vectors. The motivation is to improve memory efficiency of packet classification engine by checking various ruleset features. Even the FSBV algorithm itself doesn't depend on ruleset features; the architecture proposed [6] depends on the features of ruleset. As mentioned above, those ruleset features may not be there in all classifiers the lack of thereof will potentially yield less performance. In present work, a architecture for packet Classification is explained, where performance is independent from explained ruleset features and suitable for various hardware platforms. A Bit-Vector (BV) based approach used to represent ruleset. Each rule represented as a collection of sub-rules and an algorithm said StrideBV proposed to generate sub-rules, this is an extended version of the Field Split Bit Vector (FSBV) algorithm which is proposed in [6] the solution offers user the flexibility of deciding bit width of each sub-rule, in turn which decides the performance either of the architecture. Which yields high memory efficiency also enhances the scalability of this approach.

III. PACKET CLASSIFICATION PROCESS

The complete solution is divided in two phases,

1) StrideBV algorithm and

2) Integration of range search and modularization of Stride BV architecture.

A. Problem Statement

Packet classification is generic scheme in which a arbitrary counts of header fields of an packet can be checked for the classification purpose. The most valiantly used scheme is 5-field packet classification where the

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING

National Conference on Advanced Innovation in Engineering and Technology (NCAIET-2015)

Alva's Institute of Engineering and Technology, Moodbidri

Vol. 3,Special Issue 1,April 2015

following tuple of headers of every incoming packet is checked: Destination IP (DA), Source IP (SA), Destination Port (DP), Source Port (SP), and Protocol (PR).

The type of lookup needed for each field is different. In 5-field classification, the 2 IP address fields needs prefix match, the two port fields including protocol field needs either range or exact match. With understanding, the problem is defined as follows:

Given a packet classification ruleset that has N number of rules that considers d number of packet header fields, f_0, f_1,…, f_{d-1}, devise:

- A lookup scheme whose performance is independent of the features or properties of ruleset
- A hardware architecture to perform wire-speed packet classification for 400 Gbps and beyond

B. Stride BV algorithm

In [6] an algorithm called Field Split Bit-Vector (FSBV) is introduced which performs individual field search as chain of sub-field search, in the form bit-vectors. The motivation is to improve the efficiency of memory of the engine used for packet classification by exploiting various ruleset features. The FSBV algorithm itself doesn't depend on ruleset features the architecture proposed in [6] depends on the features of ruleset. However, as said above, such ruleset features may not be present all classifiers and the lack of it can potentially yield poor performance.

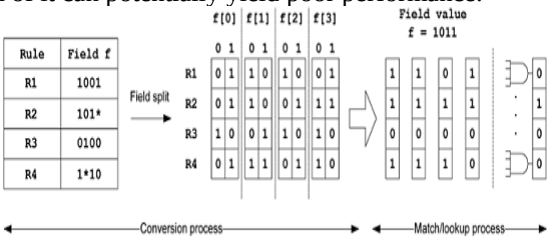


Fig. 1. Bit vector generation for FSBV and processing of header

Considering the varied nature of packet classification ruleset [5], [7], a new solution that is enables guaranteed performance for any classifier is in high demand. The FSBV algorithm proposed [6] has the characteristics being feature independent. But, the algorithm is applied only to set of selected field as memory optimization being main goal. This caused the final solution to become ruleset-feature reliant. Over here, its generalized that FSBV algorithm which extends the use of it build a ruleset-feature independent packet classification solution, named StrideBV. In the original FSBV algorithm, each W bit field was split into W 1 bit sub-fields. However, it is possible to generalize the sub-field bit length without affecting the operation of the packet classification process. In the proposed solution, a sub-field length of k bits is considered and we refer to such a sub-field as a stride. Due to this reason and the underlying data structure being bit-vector, the proposed solution is named StrideBV. We discuss the StrideBV algorithm in detail here. Each k bits of the W bit rule can perform independent matching on the corresponding k bits of an input packet header. In other words, we can divide the W bit rule into W=k of k bit sub-fields. Each sub-field corresponds to 2^k N bit-vectors: a N bit-vector for each permutation of the k bits. A wildcard (*) value in a ternary string is mapped to all bit-vectors. To match an input header with this W bit rule, each k bits of the header of the input packet will access the corresponding memory whose depth is 2^k, and return one N bit-vector. Each such k bit header lookup will produce an N bit-vector and they are bitwise ANDed together in a pipelined fashion to arrive at the matching results for the entire classifier. The memory requirement of the StrideBV approach is fixed for a classifier that can be represented in a ternary string format (i.e. a string of 0, 1 and *) and can be represented as $\Theta(2^k \times N \times W/k)$. The search time is O(W/k) since a bitwise AND operation can be done in O(1) time in hardware and O(W/k) such operations need to be performed to complete the classification process. For simplicity and clarity, we show the BV generation and the lookup process for k = 1. One caveat with StrideBV is that it cannot handle arbitrary ranges in an efficient manner. The two port fields (SP and DP) may be represented as arbitrary ranges. For example, when a certain rule needs to filter the incoming traffic belonging to one of the well-known port numbers, the rule will have in its DP field: 0-1023. In such cases, the arbitrary ranges need to be converted into prefix format where the value of each bit can be 0, 1, or * (wildcard character).

The formal algorithms for building the bit-vectors and for performing packet classification are shown in Algorithms 1 and 2, respectively. The Permutations function accepts k bits of the rule and generates a 2-dimensional array of size 2^k (height) x k (width) in which, each row value specifies the bit-vector values for the considered stride. For example, consider a stride size of k = 2 and the rule value for a particular stride is 0*. In this case, the array output by the Permutations function will be {11, 11, 01, 01} which is the match result of {00, 01, 10, 11} against rule value 0*. The value at the jth position of the array indicates the bit-vector value for the considered stride, when the input packet header has value of the binary representation of j.

C. Proposed Algorithm

Algorithm 1- Bit-vector Generation

Require: N rules each of which is represented as a W-bit ternary string: $R_n = T_n, {}_{W\_1}T_{n,W\_2}\ldots\ldots\ldots T_{n,0}$,

n = 0, 1,….., N - 1

Ensure: 2^k X W/k, N-bit vectors:

$V_{i,j} = B_{i,j,N-1}B_{i,j,N-2}\ldots\ldots B_{i,j,0}$,

i = 0; 1, . . . , W/k - 1, and j = 0, 1, . . . , 2^k - 1

1: Initialization: Vi,j to 00 ….. 0 ¥ i, j

2: for n = 0 to N -1 do {Process R_n}

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING

National Conference on Advanced Innovation in Engineering and Technology (NCAIET-2015)

Alva's Institute of Engineering and Technology, Moodbidri

Vol. 3,Special Issue 1,April 2015

3: for i= 0 to W/k -1 do
4: S [2^k][k] = Permutations (R[I * k : (i + 1) * k])
5: for j to 1 to 2^k-1 do
6: Vi,j [i * k : (i+1)* k] = S[j]
7: end for
8: end for
9: end for

Algorithm 2- Packet Classification Process
Require: A W-bit packet header: $P_{W-1}P_{W-2}$……..P_0.
Require: 2^k X W/k, N bit-vectors:
Vi, j = $B_{i,j,N-1}B_{i,j,N-2}$…….$B_{i,j,0}$
i = 0, 1, . . ., W/k - 1, and j = 0, 1, . . . 2^k-1
Require: A N bit-vector V_m to indicate match result
Ensure: N bit-vector Vm indicates all match results
1: Initialize V_m: V_m to 11 ….. 1 All rules match initially
2: for i to 0 to W/k-1 {bit-wise AND}
3: j = [P_{i*k}: P(i+1)*k)
4: V_m to V_m $V_{i,j}$
5: end for

D. Multi-Match to Highest-Priority Match.

In [6], [8], the output of lookup engine is bit-vector which indicates the rules matching for the input packet header. This is desired in environments such as Intrusion Detection Systems (IDSs) where report includes all the matches which is necessary for processing. Whereas, in packet classification, the highest priority match reported since routing is main concern. The rules of classifier are sorted in order of decreasing priority. Extracting the highest priority match from the N bit-vector helps to translates identifying the first bit position is set to 1, when traversing the bit-vector from index 0 to N - 1. This task can be easily done using a priority encoder. The priority encoder produces the result in a single cycle. However, the length of the bit vector increases, the time required also to report the highest priority match increases. This causes the entire pipeline run at a very slow clock rate for larger BV lengths, which also affects the throughput. Remedy, we introduce here is a Pipelined Priority Encoder (PPE). A PPE for a N bit-vector consist of log $_B$ N number of stages and since the work per stage is trivial, the PPE should be able to operate at very high frequencies. Here, parameter B refers to the degree of the PPE, which indicates how many times comparison is done in a particular stage. Other words, into how many partitions the bit-vector is divided into in a given stage.

E. Modular BV Approach

Since every individual element of the BV responsible for a single rule of entire ruleset, every individual bit-level operation is independent of the rest of the bits in BV. This allows partitioning the BV without effecting the operation of the BV approach. The benefit of dividing is that we no longer needs to load a N bit BV at every pipeline stage, but a N/P bit BV, where P is the number of divisions, reducing the per pipeline stage memory bandwidth required by a factor of P. Division as done based on rule priority, i.e. first N/P rules in the first partition. This makes easy the highest priority match extraction process. To explain better the importance of the undivided BV approach, we give quantitative insight using realistic example value. Consider a classifier (i.e. a ruleset) with 2000 rules and a pipelined architecture operating at 200MHz. By considering a stride value of k = 3, the pipeline length becomes [104/3] = 35. With no partitioning, each pipeline stage will request a 2000 bit wide word at a 200 million request per second rate. This translates to an on-chip memory bandwidth of 14 Tbps. On the current FPGA, having such high bandwidth is not possible. Therefore, operating frequency drops with increasing ruleset size [3], affecting the performance of packet classification engine. Each partition will produce a portion of the longest BV, referred to as sub-BV hereafter, which contains the matching result.

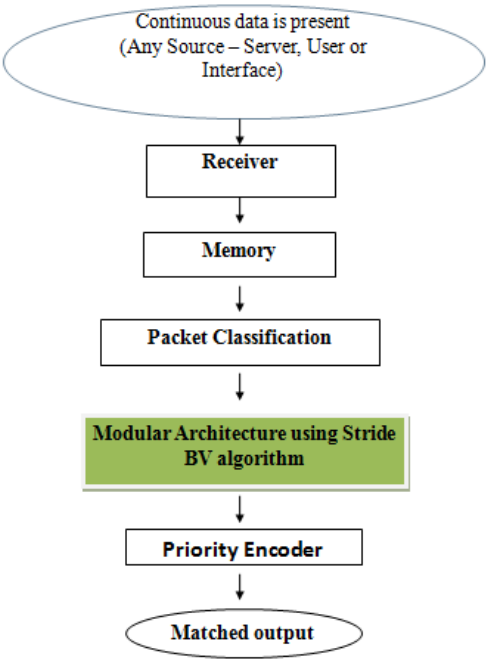


Fig. 2. Flow diagram of modular BV approach

IV. HARDWARE ARCHITECTURE

In the StrideBV architecture, in every pipeline stage, k bits of header bits of the incoming packets are used as the memory address to the every stage memory. The output bit-vector of stage memory (BVM) and the bit-vector generated from the pre stage (BVP) are bitwise ANDed both together to give the resultant bit-vector (BVR) of this stage. Same stage construction is checked throughout the pipeline. The final stride lookup stage will give output of the multi-match result and the PPE extracts the highest priority match from the resultant bit-vector. Note that BVP in stage 0 is set to all 1's to indicate that the entire ruleset is considered as potential matches.

A. Modular Architecture

The strides that process header fields with prefix and exact match are using StrideBV stages and the strides that

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING

National Conference on Advanced Innovation in Engineering and Technology (NCAIET-2015)

Alva's Institute of Engineering and Technology, Moodbidri

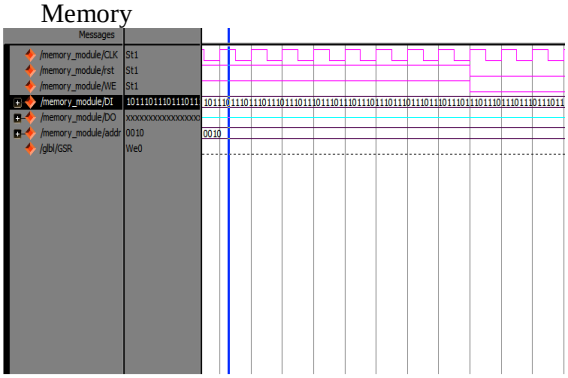
Vol. 3,Special Issue 1,April 2015

perform range comparisons are using range-search stages. The search process is followed by the priority encoder network which extracts the highest-priority match. Note that the order in which the headers are inspected does not affect the final search result. Therefore, the StrideBV stages and range-search stages can be permutated in any order the user desires. Modular architecture takes a serial search approach which consumes consecutive strides of the header as the packet progress through the pipeline stage. This leads the packet classification latency to increase linearly proportional to the header length. With longer header lengths, this can be an undesirable feature especially for applications that demand low latency operation, such as multimedia, gaming and data center environments.

V. IMPLEMENTATION AND RESULTS

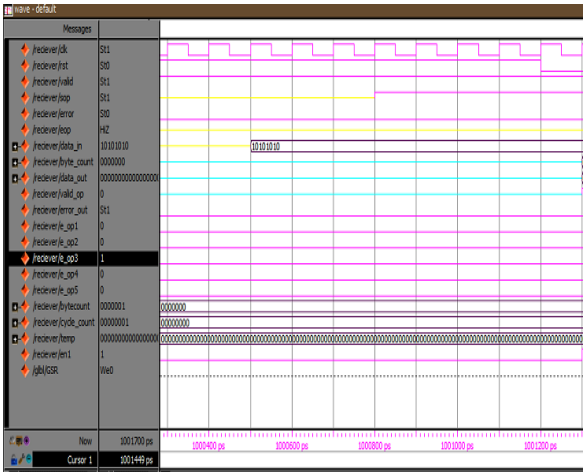
Snapshots of part of the project implemented are as given below.

A. Memory



Above snapshot shows memory unit which stores the incoming data according to address to which it has to store. The address has to be set and data is stored. Data will be written only when write enable is high.

B. Receiver



In the above snapshot, Receiver receives data only if valid bit is high and output is considered if start of packet is high (sop) and data in considers input data. There are several errors considered here such as both start of packet and end of packet can't be high. Byte count keeps counting number of bytes it receives and cycle count counts number of cycles it counts. Byte count considers error and cycle count counts if error is data occurs.

VI. CONCLUSION AND FUTURE WORK

Packet classification of various algorithms has been studied. StrideBV is algorithm used here for packet classification. This algorithm explains the classification which is done in terms of Stride which is ruleset independent compared to past methods of packet classification. The flow of modular BV approach has been discussed. Modules such as Memory and receiver are designed. Packet classification which is important segment of this work is studied and is to be implemented. The algorithm is explained and conduction work is in progress. Memory efficiency compared to previous work will be high in number and memory efficiency will be compared. In future analysis of memory efficiency can be done and compared to FSBV method which is previous version of this method.

REFERENCES

[1]. Thilan Ganegedara, Weirong Jiang, and Viktor K. Prasanna, "A Scalable and Modular Architecture for High-Performance Packet Classification," IEEE Trans on parallel and distributed systems, vol. 25, no. 5, pp. 1135-1144, MAY 2014.

[2]. T.V. Lakshman and D. Stiliadis, ''High-Speed Policy-Based Packet Forwarding Using Efficient Multi-Dimensional Range Matching,'' SIGCOMM Comput. Commun. Rev., vol. 28, no. 4, pp. 203-214, Oct, 1998.

[3]. H. Song and J.W. Lockwood, ''Efficient Packet Classification for Network Intrusion Detection Using FPGA,'' in Proc. ACM/SIGDA 13th Int'l Symp. FPGA, 2005, pp. 238-245.

[4]. W. Eatherton, G. Varghese, and Z. Dittia, ''Tree Bitmap: Hardware/ Software IP Lookups with Incremental Updates,'' SIGCOMM Comput. Commun. Rev., vol. 34, no. 2, pp. 97-122, Apr. 2004.

[5]. D. Taylor and J. Turner, ''Scalable Packet Classification Using Distributed Crossproducing of Field Labels,'' in Proc. 24th Annu. Joint IEEE INFOCOM, Mar. 2005, vol. 1, pp. 269-280.

[6]. W. Jiang and V.K. Prasanna, ''Field-Split Parallel Architecture for High Performance Multi-Match Packet Classification Using FPGAs,'' in Proc. 21st Annu. SPAA, 2009, pp. 188-196.

[7]. D.E. Taylor, ''Survey and Taxonomy of Packet Classification Techniques,'' ACM Comput. Surv., vol. 37, no. 3, pp. 238-275, Sept.2005.

[8]. Thilan Ganegedara, Viktor K. Prasanna, "StrideBV: Single Chip 400G+ Packet Classification," United States National Science Foundation undergrant No. CCF-1116781.

[9]. Miad Faezipour, Mehrdad Nourani, "Wire-Speed TCAM-Based Architectures for Multimatch Packet Classification," IEEE transactions on computers, vol. 58, no. 1, pp. 5-17, JAN 2009.