Field-Split Parallel Architecture for High Performance Multi-Match Packet Classification Using FPGAs *

Weirong Jiang
Ming Hsieh Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089, USA
weirongj@usc.edu

Viktor K. Prasanna
Ming Hsieh Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089, USA
prasanna@usc.edu

ABSTRACT

Multi-match packet classification is a critical function in network intrusion detection systems (NIDS), where all matching rules for a packet need to be reported. Most of the previous work is based on ternary content addressable memories (TCAMs) which are expensive and are not scalable with respect to clock rate, power consumption, and circuit area. This paper studies the characteristics of real-life Snort NIDS rule sets, and proposes a novel SRAM-based architecture. The proposed architecture is called field-split parallel bit vector (FSBV) where some header fields of a packet are further split into bit-level subfields. Unlike previous multi-match packet classification algorithms which suffer from memory explosion, the memory requirement of FSBV is linear in the number of rules. FPGA technology is exploited to provide high throughput and to support dynamic updates. Implementation results show that our architecture can store on a single Xilinx Virtex-5 FPGA the full set of packet header rules extracted from the latest Snort NIDS and sustains 100 Gbps throughput for minimum size (40 bytes) packets. The design achieves 1.25× improvement in throughput while the power consumption is approximately one fourth that of the state-of-the-art solutions.

Categories and Subject Descriptors

C.1.4 [**Processor Architectures**]: Parallel Architectures; C.2.0 [**Computer Communication Networks**]: General—*Security and protection*

General Terms

Algorithms, Design, Performance, Security

Keywords

FPGA, multi-match packet classification, NIDS, SRAM

1. INTRODUCTION

The power of the Internet has grown explosively to a giant open network. Unfortunately, Internet attacks require little effort and monetary investment to create, are difficult to trace, and can be launched from virtually anywhere in the world [6]. A network intrusion detection system (NIDS) [1,2] is a critical network security facility that helps protect high speed computer networks from malicious users [24]. Such systems examine network communications, identify patterns of attacks, and then take action either to terminate the connections or alert system administrators. NIDS such as Snort [2] employ thousands of rules that contain intrusion patterns. Each rule has two components: a rule header and a rule option. The rule header is a classification filter that consists of five fixed fields: protocol, source IP address, source port, destination IP address, and destination port. The rule option specifies intrusion patterns used for scanning packet payloads. The header classification results identify the related rule options that will be checked in the follow-up pattern matching. A packet may match multiple rule headers. Traditional network applications such as firewall processing require reporting only the highest-priority matching rule, which we call *best-match* packet classification [10, 13, 19, 25]. In contrast, NIDS needs *multi-match* packet classification to find all rule headers that match a given packet [23, 29, 30]. Our work focuses on the multi-match packet header classification in NIDS. Solutions for pattern matching on packet payloads [14, 17, 27, 28] are beyond the scope of this paper.

Although packet classification has been a widely studied area of research [10, 19, 25], most of the previous work has focused on best-match packet classification, while very little work has studied multi-match packet classification [30]. The explosive growth of network traffic demands multi-match packet classification must be performed in hardware. For example, the current link rate has been pushed towards 100 Gbps [20], which requires processing a packet every 3.2 ns in the worst case (where the packets are of minimum size i.e. 40 bytes). Recent work on multi-match packet classification is mostly based on ternary content addressable memories (TCAMs), which can perform fast parallel searches over all entries in one clock cycle [8, 16, 30]. However, TCAMs are expensive and not scalable with respect to clock rate, power consumption, or circuit area, compared to static random access memories (SRAMs) [13]. As the rule set size increases rapidly, alternate hardware platforms are needed for future multi-match packet classification engines.

We explore mapping state-of-the-art multi-match packet classification algorithms onto SRAM-based architectures. The

*This work is supported by the United States National Science Foundation under grant No. CCF-0702784. Equipment grant from Xilinx Inc. is gratefully acknowledged.

SPAA'09, August 11–13, 2009, Calgary, Alberta, Canada.
Copyright 2009 ACM 978-1-60558-606-9/09/08 ...$10.00.

Parallel Bit Vector (BV) algorithm [15] and its variants [5,23] are among few existing packet classification algorithms that support returning all matching rules for a packet at the same time. The BV algorithm performs the parallel lookups on each individual field of a packet header at first. The lookup on each field returns a bit vector with each bit representing a rule. A bit is set to "1" if the corresponding rule matches in this field; otherwise the bit is set to "0". The result of the bitwise AND operation on these bit vectors gives the set of rules that matches a given packet. The BV algorithm can provide high throughput at the cost of low memory efficiency. Given N rules with D fields, since the projection of the N rules on each field may have $U = O(N)$ unique values and each value corresponds to one N-bit vector, the total memory requirement of BV algorithms is at least $U * N * D = O(N^2)$ which is undesirable [10,25]. We make a key observation: If we can split a W-bit field into W subfields where each subfield takes only 1 bit, the number of unique values in each subfield will be no more than 2, i.e. the subfield value $\in [0, 1]$. Then each subfield corresponds to 2 N-bit vectors and the overall memory requirement for this W-bit field is $2W * N = O(WN)$ instead of $U * N = O(N^2)$. It will reduce the memory requirement when $2W < U$. Since W is a fixed number, the algorithm can achieve linear memory increase with the number of rules.

Field-splitting results in more subfields, leading to more bit vectors to be merged via bitwise AND operations. We exploit the reconfigurability and massive parallelism of field-programmable gate array (FPGA) technology. State-of-the-art SRAM-based FPGA devices such as Xilinx Virtex-5 [3] provide high clock rate and large amount of on-chip dual-port memory with configurable word width. It takes few milliseconds to reconfigure an entire FPGA, while the update frequency of NIDS rules is on the order of days. Thus FPGA has become an attractive platform for realizing real-time network processing engines [11,13,18,23].

Motivated by the above ideas, we propose a field-split parallel bit vector (FSBV) architecture which can be efficiently implemented in hardware. The major contributions of this paper include:

- We study the characteristics of the packet header rules from Snort which is considered as the de facto standard for NIDS [2]. We are unaware of any existing work that conducted similar systematic analysis for Snort header rules. We also believe our results can motivate more research in this area.

- Based on the Snort header rule characteristics, we propose the field-split parallel bit vector (FSBV) algorithm and apply it to the port fields to reduce the memory requirement dramatically.

- Noting that the number of unique values on the IP address fields is much smaller than twice the number of bits in these fields, i.e. $U < 2W$, we combine small TCAMs and CAMs into the architecture for matching the IP address and the protocol fields, respectively.

- Simple schemes are developed to support the unique features of Snort rule headers, including the value list, the range and the negation operators. To the best of our knowledge, this work is the first hardware implementation to support all features contained in the Snort rule headers.

- We present the detailed implementation of the FSBV architecture on a state-of-the-art FPGA. Both the dual-port BlockRAMs provided by current FPGAs and the pipelining technique are exploited to provide a high throughput of two packets per clock cycle.

- Simulation results using synthetic packet header rule sets show that the FSBV architecture achieves linear memory increase with the number of rules. Implementation results show that the FSBV architecture can store the full set of latest Snort header rules on a single Xilinx Virtex 5 FPGA using small amount of on-chip resources. It sustains 100 Gbps throughput for minimum size (40 bytes) packets.

The rest of the paper is organized as follows. Section 2 introduces the basics of Snort rules and the related work on multi-match packet classification. Section 3 studies the characteristics of Snort header rule sets. Section 4 presents the FSBV algorithm and the proposed architecture. Section 5 discusses FPGA implementation issues. Section 6 evaluates the performance of the FSBV architecture. Section 7 concludes the paper.

2. BACKGROUND

2.1 Introduction to Snort Rules

Each Snort rule is divided into two logical sections: the rule header and the rule options. The rule header contains the rule's action, protocol, source and destination IP addresses, and the source and destination port numbers. The rule option section contains alert messages and information on how the packet payload should be inspected. Figure 1 shows a sample Snort rule where the section enclosed in parenthesis is the rule option[1] and the remaining part is the rule header. In multi-match packet classification, the 32-bit source and destination IP addresses (denoted **SA**/**DA**), the 16-bit source and destination port numbers (denoted **SP**/**DP**) and the 8-bit protocol (denoted **Prtcl**) fields from the rule header of a Snort rule are matched by the input packet header. Thus we define SA, DA, SP, DP and Prtcl as the five fields of a packet header rule.

[1] Please refer to [2] for more details about Snort rule options.

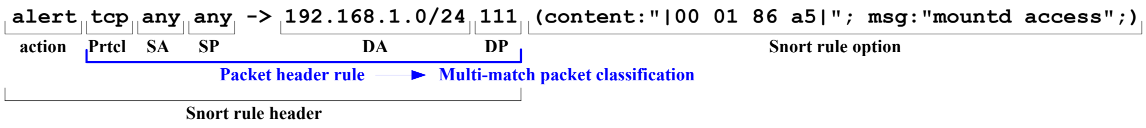

alert tcp any any -> 192.168.1.0/24 111 (content:"|00 01 86 a5|"; msg:"mountd access";)
action
Prtcl
SA
SP
DA
DP
Snort rule option
Packet header rule
Multi-match packet classification
Snort rule header

Figure 1: Interpreting a sample Snort rule

The IP addresses in SA/DA fields are specified as prefixes, which can represent either a network or a single host. The port numbers in SP/DP fields can be specified as `any`, a single number, or a range. The protocol field in the current version of Snort has only four values: `tcp`, `udp`, `icmp` and `ip`. For the SA/DA and SP/DP fields, Snort supports specifying a list of values enclosed within square brackets. Snort also provides the negation operator "!". For example, `![60,80]` indicates any port number except 60 and 80. In addition, Snort uses "EXTERNAL_NET" as an implicit negation of "HOME_NET" in SA/DA fields [29].

2.2 Prior Work

Most of the existing multi-match packet classification engines are based on TCAMs where each input performs parallel search over all entries in one clock cycle and only the first matching index is output [8,16,30]. Some early work by Lakshminarayanan et al. [16] exploits the extra bits in each TCAM entry and finds all matching rules for a packet in multiple clock cycles. The number of clock cycles needed to classify a packet is linear in the number of matching rules. This results in low speed and high power consumption.

One of the state-of-the-art TCAM-based multi-match packet classification solutions is proposed by Yu et al. [29], which is based on geometric intersection of rules. If two rules match a same packet header, several intersection rules which cover the overlap between the two rules are created and inserted into the TCAM. Only one TCAM lookup is needed to obtain all matching results. However, the number of intersection rules can be $O(N^D)$, where N is the total number of rules and D is the number of fields. Though the authors later propose the Set Splitting Algorithm (SSA) [30] to split the rule set into two groups to remove at least half of the intersections among the rules, the worst-case memory requirement is still $O(N^D)$. Thus this approach is expensive with respect to both memory and power consumption when the rule set has many intersections.

Faezipour et al. [8] propose the maximum-minimum intersection partitioning (MX-MN-IP) approach for TCAM-based multi-match packet classification. The power consumption is reduced by partitioning the entire rule set into several disjoint partitions so that only one partition is active in classifying a packet. Each partition is further partitioned so that each sub-partition needs only one clock cycle to output no more than one matching result. However, the MX-MN-IP scheme may need a large number of small TCAMs, which is not practical for real implementation. Since the number of partitions depends on the characteristic of the rule set, the power reduction ratio can vary a lot. As shown in [8], some rule sets can achieve only 5% power saving compared to conventional (non-partitioning) TCAM-based approaches.

By combining TCAMs and the original BV algorithm, Song et al. [23] present an architecture called BV-TCAM for multi-match packet classification on FPGAs. A TCAM performs prefix or exact match, while a multi-bit trie implemented in Tree Bitmap [7] is used for source or destination port lookup. It prevents range-to-prefix expansion for port fields. However, BV-TCAM still suffers from $O(N^2)$ memory requirement, for the same reason as stated in Section 1 for the BV algorithm.

3. ANALYSIS OF SNORT HEADER RULE SETS

Although there are some published observations on the characteristics of real-life packet header rule sets [4,9,19,26], most of them are used for best-match rather than multi-match packet classification. To the best of our knowledge, even though Snort header rule set is currently the largest publicly available multi-match packet classification rule set, its characteristics have not been studied.

We collected 10 rule sets from Snort website [2]. Their statistics is shown in Table 1. We counted the number of unique values for each item listed in Table 1[2]. We have the following observations:

- The number of Snort rules is much larger than the number of unique rule headers. In other words, a rule header is shared by many rules in Snort.

- The number of rule headers remains quite small, though it has increased gradually. The number of unique rule headers has almost doubled since 2005.

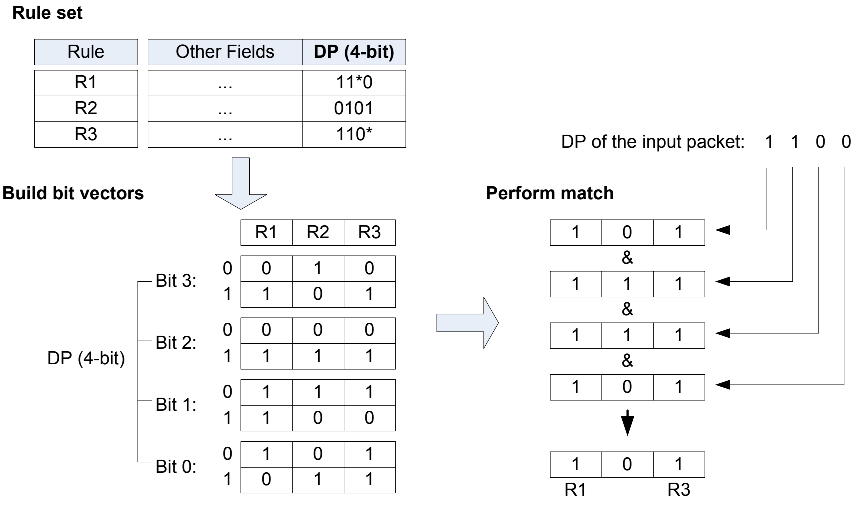
- Though the number of rule headers has been increasing gradually, the number of unique values for SA/DA fields tends to be small, less than 15.

- Unlike SA/DA fields, the number of values for SP/DP fields is increasing at a similar rate as that of rule headers. The number of unique SP/DP values is also on the same order of that of rule headers.

[2] Only the active rules (i.e. the rules not being commented) are considered.

Table 1: Statistics of Snort rule sets

Rule set	Snort version	Date	# Rules	# Rule headers	# SA	# DA	# SP	# DP	# Prtcl
R0	2.3.0	20050405	3182	**323**	11	13	87	173	4
R1	2.4.0	20050722	3462	**340**	11	13	91	183	4
R2	2.3.0	20070911	8171	**589**	10	14	198	316	4
R3	2.4.0	20070911	8346	**594**	10	14	198	320	4
R4	2.6.0	20080924	9290	**613**	10	13	203	330	4
R5	2.7.0	20081017	9244	**594**	10	13	190	327	4
R6	2.8.0	20080122	9040	**600**	10	14	202	321	4
R7	2.8.0	20080826	9277	**620**	10	13	204	336	4
R8	2.8.0	20081017	9257	**597**	10	13	187	332	4
R9	2.8.4	20090421	5662	**609**	10	13	184	344	4



Rule	Other Fields	DP (4-bit)
R1	...	11*0
R2	...	0101
R3	...	110*

		R1	R2	R3
Bit 3:	0	0	1	0
	1	1	0	1
Bit 2:	0	0	0	0
	1	1	1	1
Bit 1:	0	1	1	1
	1	1	0	0
Bit 0:	0	1	0	1
	1	0	1	1

Figure 2: Motivating example of FSBV

- The value of the protocol field is restricted to be `tcp`, `udp`, `icmp` and `ip`. Thus the number of unique values of this field remains at 4.

Moreover, we examined the usage of those unique features provided by Snort rules, including the value list, the negation operator, and the range operator for port fields.

- Most of port fields are specified as a single value. Over 85% of the unique values for SP/DP fields are specified as a single value, while only around 10% of port field values are specified as ranges.
- Negation operator is seldom used. Apart from the implicit negation by using "EXTERNAL_NET" in SA/DA fields, only SP field has two negation values.
- Each field uses few value lists. The number of value lists in SA/DA/SP/DP fields is 0/3/1/10, respectively. The value lists in SP/DP fields have no more than 4 values, while the value list in DA field contains up to 18 IP addresses.

4. ALGORITHMS AND ARCHITECTURE

4.1 The Field-Split Bit Vector (FSBV) Algorithm

As discussed in Section 1, original BV algorithms [15, 23] suffer from $O(N^2)$ memory explosion, if the number of unique values in a field (denoted U) is of the same order as the number of rules (denoted N). According to the Snort rule set statistics shown in Table 1, the large number of port field values will result in a large number of N-bit vectors, since each unique value corresponds to one N-bit vector. Moreover, we need to build a search structure, such as a quad search tree [12] or a compressed multi-bit trie [23], for an input packet to find the matching port number. Given U unique values in a W-bit port field, we need $O(U)$ and $O(UN)$ memory to store the search structure and the U N-bit vectors, respectively. The search time on this field is either $O(\log U)$ using binary (or quad) search tree or $O(W)$ using trie.

Noting that the majority of the values in the port fields are just single numbers, we can represent them in binary format. Even for ranges, we can represent them in a couple of ternary strings where the value of each bit is 0, 1, or "*" (don't care). In the following analysis, we assume a port field of a rule can be represented as a ternary string. Each bit of the W-bit port field can perform independent matching on the corresponding bit of an input packet. In other words, we can divide the W-bit field into W 1-bit subfields. Each subfield corresponds to 2 N-bit vectors: one for the bit value of 0, and the other for the value of 1. The "don't care" value in a ternary string is mapped to both bit vectors. The memory requirement becomes $2WN$. To match an input packet on this W-bit field, each bit of the field of the input packet will access the corresponding memory whose depth[3] is 2, and return one N-bit vector. Finally W N-bit vectors are bitwise ANDed to get the matching vector for this field. The search time is $O(W)$ since a bitwise AND operation can be done in $O(1)$ time in hardware.

Figure 2 shows an example of applying the FSBV algorithm for matching the DP field of a packet against three rules. The formal algorithms for building the bit vectors and for performing packet classification, are shown in Algorithms 1 and 2, respectively.

The correctness of the FSBV algorithm is proved as follows. Consider N rules. Each rule contains a W-bit field which is represented as a ternary string. Use Algorithms 1 and 2 to build the $2W$ N-bit vectors and to classify a packet, respectively. Then,

THEOREM 4.1. *An input W-bit number α matches the W-bit field of the ith rule for any $i \in \{0, 1, \cdots, N-1\}$ if and only if the ith bit of the final N-bit matching vector is set.*

PROOF. An input W-bit number α matches the W-bit field of a rule, if and only if each bit of α matches the corresponding bit of the field. This is guaranteed since the field is represented as a ternary string.

[3] Throughout this paper, *depth* of a memory is defined as the number of entries in it. The *width* of a memory is defined as the number of bits in each entry.

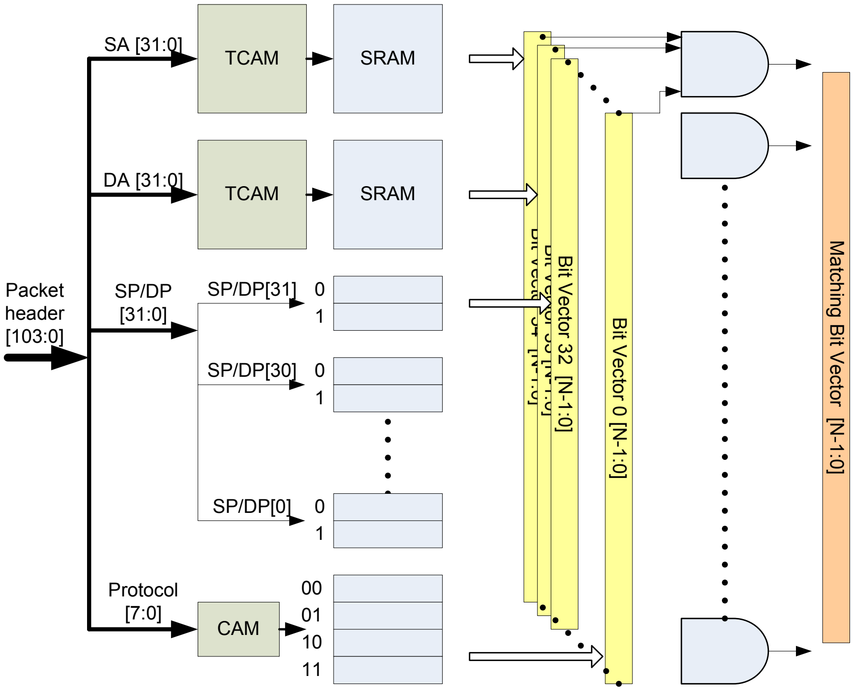


Figure 3: Basic architecture of FSBV

```
Algorithm 1 Building bit vectors
Input: $N$ rules each of which is represented as a $W$-bit ternary string: $R_i = T_{i,W-1}T_{i,W-2}\cdots T_{i,0}$, $i = 0,1,\cdots,N-1$.
Output: $2W$ $N$-bit vectors: $V_i = B_{i,N-1}B_{i,N-2}\cdots B_{i,0}$, $i = 0,1,\cdots,2W-1$.
Initialization: $V_i \leftarrow 00\cdots 0$, $i = 0,1,\cdots,2W-1$.
for $i \leftarrow 0$ to $N-1$ do {Process $R_i$}
    for $j \leftarrow 0$ to $W-1$ do
        if $T_{i,j} == *$ then
            $B_{2j,i} \leftarrow 1$
            $B_{2j+1,i} \leftarrow 1$
        else {$T_{i,j} == 0$ or $1$}
            $B_{(2j+T_{i,j}),i} \leftarrow 1$
            $B_{(2j+1-T_{i,j}),i} \leftarrow 0$
        end if
    end for
end for
```

```
Algorithm 2 Classifying a packet
Input: A $W$-bit packet header: $P_{W-1}P_{W-2}\cdots P_0$.
Input: $2W$ $N$-bit vectors: $V_i = B_{i,N-1}B_{i,N-2}\cdots B_{i,0}$, $i = 0,1,\cdots,2W-1$.
Output: A $N$-bit vector $V_p$ indicating all matching rules.
Initialize a $N$-bit vector: $V_p \leftarrow 11\cdots 1$.
for $i \leftarrow 0$ to $W-1$ do {bit-wise AND}
    $V_p \leftarrow V_p$ & $V_{2i+P_i}$
end for
```

Also, each bit of α will fetch a N-bit vector where the ith bit is set if and only if it matches the corresponding bit of the field of the ith rule.

Note that the ith bit of the final matching vector is set if and only if all W N-bit vectors fetched by α have the ith bit set.

Thus, the theorem is proved. □

Note that the FSBV algorithm achieves memory efficiency when $U > 2W$, i.e., the number of unique values of a field is larger than twice the number of bits of the field. According to the Snort rule set characteristics, applying the FSBV algorithm onto SP/DP fields will reduce the memory consumption dramatically, compared with previous BV algorithms [15, 23]. Also note that, the total number of bit vectors in the FSBV algorithm is bounded by the total number of bits in the packet header. The bounded parallelism in FSBV makes a major distinction from TCAMs where massive parallelism over all entries results in high power consumption and poor scalability [25].

4.2 Basic Architecture

The FSBV algorithm can be mapped easily onto hardware architectures. Figure 3 shows the basic FSBV architecture for matching N rules each of which can be represented as a ternary string. According to the characteristics discussed in Section 3, the number of unique values in SA/DA/protocol fields is much smaller than the number of bits of these fields. Thus, we use TCAMs and CAMs which are efficient in matching the input with a small number of entries for these fields. The small TCAMs perform prefix matching on SA/DA fields, while a 4-entry CAM performs exact matching on protocol field.

In this architecture, all bit vectors are stored in SRAMs. For SA/DA/protocol fields, the number of entries stored in SRAMs is equal to that in TCAM/CAMs. SP/DP fields have $16 + 16 = 32$ bits in total, and need 32 memories each of which has a depth of 2 to store the bit vectors. Given one input packet, 3 and 32 N-bit vectors will be output

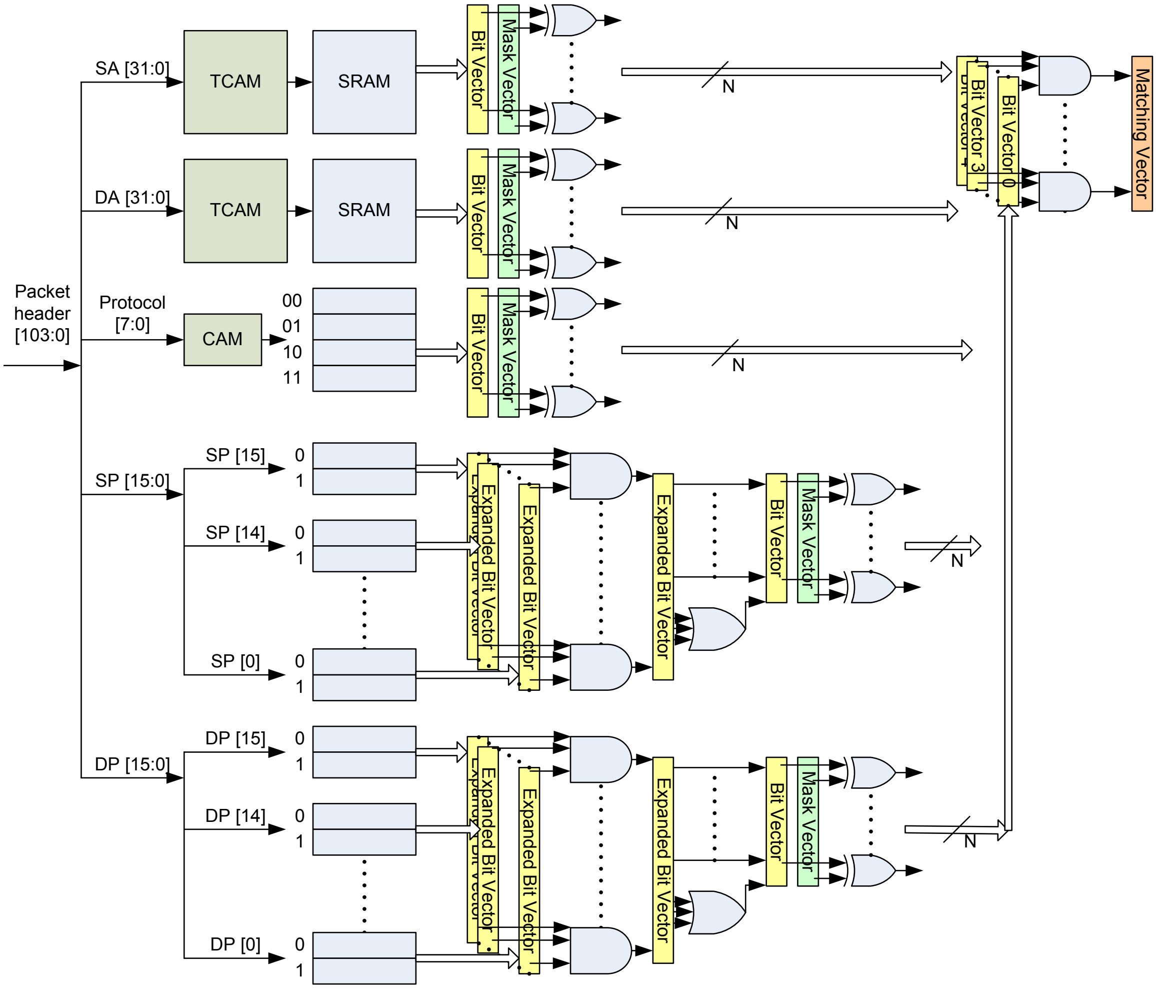


Figure 4: FSBV architecture supporting Snort rule features

from SA/DA/protocol fields and SP/DP fields, respectively. These 35 N-bit vectors will be merged into the final N-bit matching vector by bitwise AND operations in hardware.

4.3 Supporting Snort Features

As discussed in Section 2.1, Snort rules provide several unique features including range and negation operators and value list. The FSBV architecture can support them even though these features are not widely used in current Snort rules. To the best of our knowledge, no hardware implementation supporting these features is known in the literature.

4.3.1 Range

We need to convert a range into ternary strings to satisfy the assumption of the FSBV algorithm that each rule is represented as a ternary string. Unlike most of the previous TCAM-based solutions which convert a W-bit range into up to 2(W-1) prefixes [16], we adopt the algorithm proposed recently by Sasao [21] where a W-bit range can be converted into no more than 2(W-2) ternary strings. Then we use the same method as for the value list (shown later) to implement ranges. The range-to-string conversion may increase the memory requirement. In the worst case, when all port fields of the N rules are specified as ranges, the memory requirement will be $O(W^2N)$ rather than $O(WN)$. Fortunately the characteristics shown in Section 3 prevents such worst case, and our evaluation results in Section 6 demonstrate that the memory increase due to range-to-string conversion is relatively low.

4.3.2 Negation

Section 3 shows that the negation operator is seldom used in current Snort rule sets. However, a negated value can be converted into multiple ternary strings. This affects the memory efficiency dramatically [29]. Some TCAM-based work [29] reorganizes the rule set and inserts extra rules to remove negation. We propose a simple method which does not add any new rule. Given N rules and D fields, we assign a N-bit mask vector for each field. Initially all bits of a mask vector is reset. If the ith rule ($i = 0, 1, \cdots, N-1$) has negated value in its jth field ($j = 0, 1, \cdots, D-1$), the ith bit of the mask vector for the jth field is set.

For a field with negation, we use its unnegated value to build bit vectors. When performing packet classification, we merge the bit vectors of each field independently, and obtain D partial matching vectors. The partial matching vector from the jth field ($j = 0, 1, \cdots, D-1$) is bitwise XORed with the mask vector of the jth field. If the ith bit of the mask vector is set, the matching result for the ith rule on this field will be negated. For Snort rule sets, only 3 N-bit mask vectors are needed for SA/DA/SP fields, which can be easily integrated into the FSBV architecture.

4.3.3 Value List

Snort allows SA/DA/SP/DP fields to be specified using a list of values. As shown in Section 3, current Snort rule set uses few value lists. We can simply expand a value list to multiple values. However, the range-to-string conversion also results in a list of values for a single field. If one rule has multiple fields where each field is specified as a value list, this rule may be expanded to $\prod_{j=1}^{D} M_j$ rules, where D is the number of fields and M_j is the number of values in the list of the jth field.

To solve the above problem, we propose a simple method called *expansion-aggregation*. For example, if a field of the ith rule is specified as a list of M values, the ith bit of all bit vectors for this field is expanded to M bits. When building the bit vectors for this field, each single value out of the M values is treated as a rule. To perform packet classification, after all bit vectors of this field are bitwise ANDed to a single $(N+M-1)$-bit vector, the M bits corresponding to the M values are ORed to result in one bit. Thus the output partial matching vector of this field is still a N-bit vector.

4.3.4 Putting It All Together

After integrating the above techniques into the basic FSBV architecture, we obtain the FSBV architecture supporting all the features of Snort rules, as shown in Figure 4.

5. FPGA IMPLEMENTATION

FPGA technology has become an attractive choice for implementing real-time network processing engines [13,23], due to its ability to quickly reconfigure and to offer massive parallelism. Considering the features of FPGA, several design issues must be addressed for implementing the FSBV architecture on FPGAs.

5.1 Blocking

According to Figure 4, the FSBV architecture needs a large amount of wide memories with depth of 2, to store the bit vectors for SP/DP fields. However, the majority of on-chip memory of FPGAs is organized in blocks. For example, the minimum size block memory on Xilinx Virtex 5 FPGAs is 18 Kb, programmable from 16K×1 bit to 512×36 bits [3]. In other words, the minimum memory depth is 512, which requires the address width be at least 9 bits. To exploit the large amount BlockRAMs provided by current FPGAs, we propose a blocking scheme which considers B bits instead of 1 bit as a subfield inside a W-bit field. After blocking, there are $\lceil \frac{W}{B} \rceil$ subfields each of which consumes $2^B N$ bits, given N rules. The total memory requirement is $2^B N \frac{W}{B} = \frac{2^B}{B} WN$, while we need $\frac{W}{B}$ bitwise AND operations to merge the bit vectors to get the partial matching vector of this field. In our FPGA implementation, $B = 9$, equal to the minimum address width of BlockRAMs on Xilinx FPGA.

Blocking provides a flexible way to control the degree of parallelism in the FSBV architecture. A larger B results in fewer N-bit vectors to be merged at the cost of increased memory consumption. Such controllable parallelism makes the FSBV architecture distinct from other solutions including the BV algorithm which uses limited parallelism and TCAMs which offer massive parallelism.

5.2 Pipelining

Given N rules, one input packet fetches 3 N-bit vectors from SA/DA/protocol fields, through TCAM/CAM access. For SP/DP fields with the blocking size of $B = 9$, $\lceil \frac{32}{9} \rceil = 4$ N-bit vectors are output from SRAMs. However, it results in low clock frequency if these $3 + 4 = 7$ N-bit vectors are bitwise ANDed in one (wide) clock cycle.

We employ pipelining to minimize the routing delay to achieve high clock rate. The pipeline consists of 7 stages: 2 stages for SP, 2 stages for DP, 1 stage for SA, 1 stage for DA, and 1 stage for protocol. At each stage, some bits of the input packet header access the local memory of that stage to retrieve the corresponding bit vector, which is then bitwise ANDed with the bit vector carried from the previous stage. The resulting bit vector is carried to the next stage along with the packet header. By further exploiting the dual-port BlockRAMs provided by Xilinx FPGA, we can input two packets every clock cycle so that the throughput is doubled.

6. PERFORMANCE EVALUATION

We conducted various experiments to evaluate the performance of the FSBV algorithm and the architecture, using both synthetic and real-life Snort rule sets.

6.1 Results on Synthetic Rules

Note that the Snort rule header set is fairly small; we are not aware of any larger publicly available multi-match packet classification rule set. To evaluate the scalability of the FSBV algorithm, we conducted experiments using randomly generated rule sets with various sizes as in [8, 30]. For a fair comparison, these synthetic rules did not contain the unique features of Snort rule sets, but had the similar characteristics as Snort header rule sets, e.g. the number of unique values in each field.

We compared the FSBV algorithm with the BV and the HyperCuts [22] algorithms and TCAM-based solutions, with respect to the average memory requirement per rule. The HyperCuts algorithm is considered as one of the most scalable algorithms for best-match packet classification [13]. According to Figure 5 where logarithmic (base 10) scale is used for Y-axis, the FSBV algorithm achieved linear memory increase with the number of rules. Compared to TCAMs which consumed 13 bytes per rule, the FSBV algorithm consumed less than 12 bytes per rule.

6.2 Results on Snort Rules

We mapped the 620 header rules from R7 (the largest Snort header rule set shown in Table 1) onto the FSBV architecture. The CAM for protocol field needed 4 entries and the TCAM for SA field needed 11 entries. For DA field, there were some value lists containing up to 18 IP addresses. After expansion, the TCAM for DA field needed 53 entries. Each bit of the 16-bit SP field corresponded to two expanded 649-bit vectors, while each bit of DP field corresponded to two expanded 668-bit vectors. Also there were three 620-bit mask vectors for SA/DA/SP fields. Thus it needed $4*620 + 11*620 + 53*620 + 16*2*649 + 16*2*668 + 3*620 = 86164$ bits to store all the bit vectors. The average memory size per rule was 17.4 bytes including the memory overhead for supporting the unique features in Snort. The overhead is

Table 2: Performance comparison

Approach	Platform	Storage	Throughput	Power	Support for Snort features
BV-TCAM [23]	FPGA	73.8 bytes/rule	10 Gbps	17.7 μW/rule	No
TCAM-SSA [30]	ASIC	13 bytes/rule	20 Gbps	312 μW/rule	No
MX-MN-IP [8]	ASIC	13 bytes/rule	80 Gbps	296 μW/rule	No
FSBV	FPGA	17.4 bytes/rule	100 Gbps	4.2 μW/rule	Yes

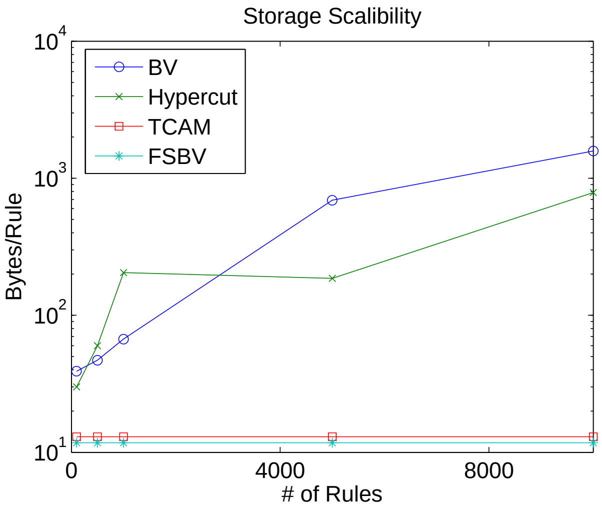


Figure 5: Comparing FSBV with other solutions.

low, less than 6 bytes per rule. In contrast, the BV-TCAM architecture [23] needed 58830 bits to store all the bit vectors for just 222 rules, i.e., 33 bytes per rule, excluding the large amount of memory needed to store the trie structures for matching port fields.

We implemented the FSBV architecture on a Xilinx Virtex-5 XC5VFX200T device with -2 speed grade. Post place and route results from Xilinx ISE 10.1 development tools showed that the FSBV architecture achieved a clock frequency of 167 MHz, while consuming only 7% of the slices and 13% of the Block RAMs. Due to the use of dual-port RAMs, the architecture could process two packets every clock cycle, resulting in a high throughput of 100 Gbps for minimum size (40 bytes) packets.

Table 2 compares the FSBV architecture with the state-of-the-art solutions for multi-match packet classification. The FSBV architecture achieved low memory requirement which was comparable to TCAMs and much less than BV-TCAM. Note that none of the BV-TCAM [23], the TCAM-SSA [30] and the MX-MN-IP [8] can handle the negation and value list problems. Much higher memory consumption can be expected in these solutions for Snort rules. For both TCAM-SSA and MX-MN-IP for which the performance depends highly on the rule set, we considered the best case and assumed that each rule could be stored as one entry in TCAM without any redundancy.

The throughput results of the FSBV and the BV-TCAM architectures were based on FPGA implementations, while those of the TCAM-SSA and the MX-MN-IP were based on application-specific integrated circuit (ASIC) implementations which usually have a much higher clock rate than FPGAs. We considered the clock frequency of TCAMs to be 250 MHz [8,30]. The BV-TCAM paper [23] did not present the actual throughput results of the implementation. We used the predicted value given in [23]. According to Table 2, the FSBV architecture achieved much higher throughput than state-of-the-art solutions.

We estimated the power consumption of SRAM and TCAM based on the data from [25]. The power consumption per bit of TCAMs is on the order of 3 μW, while that for SRAMs is less than 30 nW [25]. Considering the 5% power saving, the power consumption of MX-MN-IP was calculated as 95% of the result of the TCAM-SSA. As two SRAM-based architectures, the FSBV and the BV-TCAM solutions are much more power-efficient than the TCAM-based approaches (TCAM-SSA and MX-MN-IP). Due to lower memory requirement, the FSBV architecture achieved 4-fold reduction in power consumption over BV-TCAM.

7. CONCLUSION AND FUTURE WORK

Multi-match packet classification is a critical function in NIDS. This paper proposed a novel field-split parallel bit vector (FSBV) architecture for multi-match packet classification. This approach requires linear memory increase with the number of rules. After studying the characteristics of Snort NIDS rule sets, we combined into the architecture small TCAMs and CAMs, which were used for matching IP address and protocol fields, respectively. FPGA implementation results showed that the FSBV architecture could store the full set of the current Snort rule headers using small amount of on-chip resources and sustains 100 Gbps throughput for minimum size (40 bytes) packets.

In the future, we plan to measure more precisely the power consumption of the FSBV architecture in both ASIC and FPGA implementations. The impact of dynamic updates on the throughput will be evaluated under real-life traces. Future work also includes integrating the FSBV architecture with pattern matching engines [27] for deep packet inspection.

8. REFERENCES

[1] Bro intrusion detection system. *http://www.bro-ids.org.*

[2] Snort: the de facto standard for intrusion detection/prevention. *http://www.snort.org.*

[3] Xilinx virtex-5 multi-platform FPGA. *www.xilinx.com/products/virtex5/.*

[4] F. Baboescu, S. Singh, and G. Varghese. Packet classification for core routers: Is there an alternative to CAMs? In *INFOCOM '03: Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 53–63. IEEE, March/April 2003.

[5] F. Baboescu and G. Varghese. Scalable packet classification. *IEEE/ACM Trans. Netw.*, 13(1):2–14, Feb. 2005.

[6] R. Bajcsy, T. Benzel, M. Bishop, B. Braden, C. Brodley, S. Fahmy, S. Floyd, W. Hardaker, A. Joseph, G. Kesidis, K. Levitt, B. Lindell, P. Liu, D. Miller, R. Mundy, C. Neuman, R. Ostrenga, V. Paxson, P. Porras, C. Rosenberg, J. D. Tygar, S. Sastry, D. Sterne, and S. F. Wu. Cyber defense technology networking and evaluation. *Commun. ACM*, 47(3):58–61, March 2004.

[7] W. Eatherton, G. Varghese, and Z. Dittia. Tree bitmap: hardware/software IP lookups with incremental updates. *SIGCOMM Comput. Commun. Rev.*, 34(2):97–122, 2004.

[8] M. Faezipour and M. Nourani. Wire-speed TCAM-based architectures for multimatch packet classification. *IEEE Trans. Comput.*, 58(1):5–17, Jan. 2009.

[9] P. Gupta and N. McKeown. Packet classification on multiple fields. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 147–160. ACM, 1999.

[10] P. Gupta and N. McKeown. Algorithms for packet classification. *IEEE Network*, 15(2):24–32, March/April 2001.

[11] G. S. Jedhe, A. Ramamoorthy, and K. Varghese. A scalable high throughput firewall in FPGA. In *FCCM '08: Proceedings of the 16th International Symposium on Field-Programmable Custom Computing Machines*, pages 43–52. IEEE Computer Society, 2008.

[12] W. Jiang and V. K. Prasanna. A FPGA-based parallel architecture for scalable high-speed packet classification. In *ASAP '09: Proceedings of the International Conference on Application-Specific Systems, Architectures and Processors*, July 2009.

[13] W. Jiang and V. K. Prasanna. Large-scale wire-speed packet classification on FPGAs. In *FPGA '09: Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays*, pages 219–228. ACM, 2009.

[14] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. S. Turner. Algorithms to accelerate multiple regular expressions matching for deep packet inspection. In *SIGCOMM '06: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications*, pages 339–350. ACM, 2006.

[15] T. V. Lakshman and D. Stiliadis. High-speed policy-based packet forwarding using efficient multi-dimensional range matching. *SIGCOMM Comput. Commun. Rev.*, 28(4):203–214, 1998.

[16] K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary. Algorithms for advanced packet classification with ternary CAMs. In *SIGCOMM '05: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications*, pages 193–204. ACM, 2005.

[17] P.-C. Lin, Y.-D. Lin, T.-H. Lee, and Y.-C. Lai. Using string matching for deep packet inspection. *Computer*, 41(4):23–28, April 2008.

[18] A. Nikitakis and I. Papaefstathiou. A memory-efficient FPGA-based classification engine. In *FCCM '08: Proceedings of the 16th International Symposium on Field-Programmable Custom Computing Machines*, pages 53–62. IEEE Computer Society, 2008.

[19] L. Qiu, G. Varghese, and S. Suri. Fast firewall implementations for software and hardware-based routers. In *ICNP '01: Proceedings of the Ninth International Conference on Network Protocols*, pages 241–250. IEEE Computer Society, 2001.

[20] B. Reed. Verizon moving to 100Gbps network in '09. *http://www.networkworld.com/news/2008/031008-verizon-100gpbs-network.html*.

[21] T. Sasao. On the complexity of classification functions. In *ISMVL '08: Proceedings of the 38th International Symposium on Multiple Valued Logic*, pages 57–63. IEEE, May 2008.

[22] S. Singh, F. Baboescu, G. Varghese, and J. Wang. Packet classification using multidimensional cutting. In *SIGCOMM '03: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications*, pages 213–224. ACM, 2003.

[23] H. Song and J. W. Lockwood. Efficient packet classification for network intrusion detection using FPGA. In *FPGA '05: Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays*, pages 238–245. ACM, 2005.

[24] L. Tan and T. Sherwood. A high throughput string matching architecture for intrusion detection and prevention. In *ISCA '05: Proceedings of the 32nd annual international symposium on Computer Architecture*, pages 112–122. IEEE Computer Society, 2005.

[25] D. E. Taylor. Survey and taxonomy of packet classification techniques. *ACM Comput. Surv.*, 37(3):238–275, 2005.

[26] D. E. Taylor and J. S. Turner. Scalable packet classification using distributed crossproducing of field labels. In *INFOCOM '05: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 269–280. IEEE, March 2005.

[27] Y.-H. Yang, W. Jiang, and V. K. Prasanna. Compact architecture for high-throughput regular expression matching on FPGA. In *ANCS '08: Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pages 30–39. ACM, 2008.

[28] F. Yu, Z. Chen, Y. Diao, T. V. Lakshman, and R. H. Katz. Fast and memory-efficient regular expression matching for deep packet inspection. In *ANCS '06: Proceedings of the ACM/IEEE symposium on Architecture for networking and communications systems*, pages 93–102. ACM, 2006.

[29] F. Yu, R. H. Katz, and T. V. Lakshman. Efficient multimatch packet classification and lookup with TCAM. *IEEE Micro*, 25(1):50–59, 2005.

[30] F. Yu, T. Lakshman, M. Motoyama, and R. Katz. Efficient multimatch packet classification for network security applications. *IEEE Journal on Selected Areas in Communications*, 24(10):1805–1816, Oct. 2006.