

# Efficient Packet Matching for Gigabit Network Intrusion Detection using TCAMs

Ming Gao

*Systems Engineering Institute, Xi'an Jiaotong University*  
*mgao@sei.xjtu.edu.cn*

Kenong Zhang, Jiahua Lu

*School of Electrical Engineering, Xi'an Jiaotong University*

## Abstract

*Ternary content-addressable memories (TCAMs) have gained wide acceptance in the industry for storing and searching patterns in routers. But two important problems block the way to deploy TCAMs as deep package matching engines for network intrusion detection systems (NIDS): long patterns matching and range patterns matching. A novel high speed long patterns matching architecture using cascade TCAMs for large signature set based NIDS is presented in this paper. Systems to handle tens of thousands of signatures with thousands of bytes length each can be built on such architecture. A novel efficient header rules matching system is proposed in this paper. This scheme offloads the range matching task to efficient specialized comparing engines in FPGAs so that it solves the range matching problem with high throughout performance, about 250 million packets per second (Mpps) theoretically.*

## 1. Introduction

Compared with end-host based solutions, NIDS can respond and be updated faster. The time they could save is much important for prevention, especially when the network is under the attacks from new worms. More and more NIDS have been deployed in backbone links and different kinds of Ethernets. However, most software based NIDS can not handle with high speed network today, such as the Gigabit Ethernet and ATM network because of their poor throughout performances.

SNORT [1], a popular open source light-weight NIDS, has been always used as the reference for high speed NIDS design all over the world. Two main bottlenecks block the way to deploy NIDS like SNORT into gigabit networks. One is the TCP packets parsing and flow reconstruction. Fortunately, network interface card (NIC) with TCP offload engine (TOE) chip has been commercially available now, which can handle the TCP/IP traffic processing task at data rates of multi-gigabit. The other one is multi-pattern matching using large signature set at gigabit line rates,

which is a compute-intensive task. The current SNORT set has more than 2000 signatures and nearly 300 header rules, which make up thousands of complete rules with 42 options. Such a large and complicated pattern set makes it hard for pure software based pattern matching algorithms to keep up with line rates. In fact, SNORT system on software can handle link rates no more than 100Mbps [2] under normal traffic links and worst case performance is even less. During the recent years, many research groups have raised different multi-pattern matching methods on FPGAs. Some of them can speed up the multi-pattern matching to multi-gigabit rates for SNORT set, however, with the cost of several million gates scale FPGAs.

Ternary Content Addressable Memory (TCAM) is a type of memory device that can perform search at high speeds. Given an input string, it compares the string against all entries in its memory in parallel, and reports the first entry that matches the input within a deterministic lookup time for any input. Each cell in a TCAM can take one of the three states: 0, 1 or '\*' (do not care). The 'do not care' state and deterministic matching speed make TCAM very suitable for matching variable prefix CIDR IP addresses and the SNORT signatures with the 'No Case' option.

Compared with FPGA solutions, multi-pattern matching engine using TCAM has obvious advantages in capacity, deterministic speed, patterns updating efficiency and price. However, several well-known architecture limitations make TCAM unsuitable to be used in pattern matching problems directly. Particularly, patterns with long length and range fields are very common in signature set and header rules set of SNORT system respectively. TCAM can not handle the matching task with such two kinds of patterns. In this paper, novel algorithms are developed to use TCAMs to achieve high speeds while not being restricted to these limitations.

The remainder of this paper is organized as follows: related work and their limitations are reviewed in Section 2. Multi-pattern matching problem statement and necessary definitions are in Section 3. Section 4 and section 5 present the schemes to solve the long patterns problem and range matching problem

respectively. There are the performance analysis and evaluation for the schemes in Section 6 and conclusion in Section 7.

## 2. Related work

### 2.1. Software-based methods

NIDS scan packet payloads with signatures to detect malicious intrusions. This is a string matching procedure. The most well-known software-based algorithms are: Knuth-Morris-Pratt (KMP), Boyer-Moore (BM), Aho-Corasick (AC) and Commentz-Walter (CW).

The KMP and BM algorithms are designed for single pattern searching. If the pattern length is  $m$  bytes, then it will take  $O(m+n)$  time to finish the search in an  $n$  bytes packet. If there are  $k$  patterns, the search time will be  $O(k(m+n))$ ; linearly to  $k$ . String matching module using BM algorithm is called in current SNORT system.

The AC and CW algorithms are designed for multi-pattern matching by pre-processing the patterns and building a finite automaton which can process an input packet with  $n$  bytes length in  $O(n)$  time. However, the exponential state explosion cost too much space.

Recently, lots of improved algorithms have been raised, such as the AC BM algorithm [3] and so on. All these new algorithms can improve the pattern matching speed greatly, but it is still too slow of software-based SNORT system to meet the throughput speed goals for the high speed networks.

### 2.2. FPGA-based methods

It is one of the hot research topics to accelerate NIDS using FPGAs. Most of the related works focus on the high speed deep packet payload matching problem.

Although solutions on FPGAs can reach very high throughput speeds, they indeed cost too many hardware resources. For instance, Virtex2-6000 is a 6 million gates level FPGA device. And it is impossible to reach 95 percents utilization rate in a real FPGA design. In fact, whether it is feasible and necessary to put the whole SNORT signature set into one FPGA chip, is still under discussion now. Firstly, SNORT, which is just a light-weight NIDS, has already cost so many hardware resources in FPGA solutions. The design scale would be incredible for larger NIDS signature set. And there might be even no device available for such a huge design. Secondly, when the design scale grows with the signature set, the interconnecting latency will rise and both the system operation clock frequency and throughput speed will

**Table 1. Summary of High Speed Content Matching Methods on FPGAs**

Group and Component	Device	Logic Cells	Throughput (Gbps)
Crete Pre-decoded CAMs [4]	Virtex 2-6000	64,268 (95%)	9.7
GaTech Decoder Trees [5]	Virtex 2-8000	54,890 (81%)	7
UCLA Packet Filters [6]	Spartan 3-2000	15,202 (37%)	3.2
WashU Bloom Filters [7]	Virtex 4-100	35,850 (85%)	20.4

decrease. Last, the larger the design is, the more time the updating or reconfiguration procedures cost. So systems would be quite vulnerable during the updating time, especially when the new worms are spreading.

### 2.3. TCAM-based methods

The main characters of TCAM device are as follows: Firstly, given an input string, it can report only one matching result in  $O(1)$  time. In this paper, it is resumed the use of the widely-adopted first-match TCAM, which gives out the lowest index match of the input string if there are multiple matches. Secondly, for most TCAM, a single chip can only configured to one fixed data width once, no more than 64 bytes in current state of arts. Thirdly, each pattern takes one entry, and every bit can be set to three states including the 'do not care' state. Fourthly, it is technically hard to expand the capacity of a single TCAM chip. It can reach 18M bits in current state of arts. The deepest configuration is 4 bytes width with 256k entries and the widest configuration is 64 bytes width with 32k entries. Last, in particular configurations (e.g., 8 bytes or 16 bytes width), it can perform one search cycle in less than 4 ns (266 million searches per second).

Most related works focus on efficient methods on packet classifier using TCAM for very high speed networks. Their proposals are built on the same idea that they use FPGA or SRAM to help TCAM solve the efficient range matching problem.

Fang Yu et al. from UC Berkeley proposed their TCAM based scheme to handle complex patterns, such as arbitrarily long patterns, correlated patterns and patterns with negation, using accessorial SRAM [8]. But large and complicated data tables need to be maintained in SRAM and frequent memory accessing operations need to be taken to update the tables. There are still some problems with both the performance and function of this scheme in practice.

## 3. Problem definition

The syntax of the SNORT rules takes the form like expression (1) below. The HID, CID and RID stand for header ID, content ID and rule ID respectively. The most compute-intensive task for SNORT system is to scan the headers and payloads of packets at high line rates. Unfortunately, because of its architectural limitations, TCAM can not directly used to scan payload for the CIDs of long patterns. And it turns out space inefficiency when header patterns with range fields of ports are stored in TCAM.

$$HID \wedge (CID_0 \oplus option_0) \wedge \dots \wedge (CID_n \oplus option_n) \Rightarrow RID \Rightarrow Action \quad (1)$$

### 3.1. Long pattern problem

The long pattern is a kind of the simple string patterns. A simple pattern P of k bytes can be written as  $P=B_1B_2\dots B_k$ , where each  $B_i$  represents a byte. A signature set S is composed of n pieces of simple patterns,  $\{P_1, P_2, \dots, P_n\}$ . Assume that the lengths of the patterns in set S vary within the range  $[K_{min}, K_{max}]$ , while the width of the TCAM can only reach up to  $L_{max}$  bytes, the situation of  $K_{max} > L_{max}$  is the situation which this paper concentrates on. For instance, in the current state of arts, the  $L_{max}$  is no more than 64 bytes, while the  $K_{max}$  of SNORT signature set is 122 bytes now.

### 3.2. Range matching problem

A full-featured SNORT rule can be separated into a header part and a content part. A header rule can be represented as a 5-tuple string like {Source IP Address = any, Destination IP Address = HOME\_NET, Protocol = ICMP, Source Port > 1024, Destination Port = 23}. It turns out that the ports fields are usually defined as ranges, which can not be stored in TCAM directly. Although a range can be converted into a series of prefixes, this processing can greatly expand the rule set size. In the worst case, a sub-range of a k bits field can be converted to  $2^{(k-1)}$  prefixes. The numbers of expansions is multiplied to be up to  $4^{(k-1)^2}$  when both the source port and destination port are represented as ranges in headers. Take the worst case for instance, when  $k = 16$ , a single rule may be expanded to 900 entries in TCAM.

## 4. Long pattern matching solution

The popular method using TCAM for pattern matching is to use accessory RAM to record the historical matching information of partial pieces of long patterns. It has been abandoned because of its

large complex structure and poor performance in the worst case. A more efficient solution based on cascade TCAMs is represented as follows:

### 4.1. Configuration and work flow for the front TCAM ("TCAM\_1" in Fig. 1):

Towards different optimization goals, the TCAM\_1 can be configured into different data widths. The width can be set to  $L_1 = 8$  or 16 bytes, so that the TCAM can reach the shortest searching latency (less than 4 ns). And the longest data width can be reached when  $L_1 = 64$  bytes. In Fig. 1,  $L_1 = 4$  bytes.

Short patterns are stored into TCAM\_1 directly. Patterns shorter than  $L_1$  will be padded with "\*" (do not care) bits. Long patterns are cut into short pieces and stored into TCAM. In Fig. 1, the long pattern "ABCDEFGH" is cut into "ABCD" and "EFGH". Because "EFGH" is shorter than 4 bytes, it is padded with "\*" into "EFGH\*". Similarly, "EFGHIJKLAB" is cut into "EFGH", "IJKL" and "AB\*\*".

Because a TCAM only reports the first matching result if there are multiple matches and all matching patterns should be identified, patterns in TCAM should be organized according to their lengths in descending order. In Fig. 1, "EFGH\*" is stored in a higher index

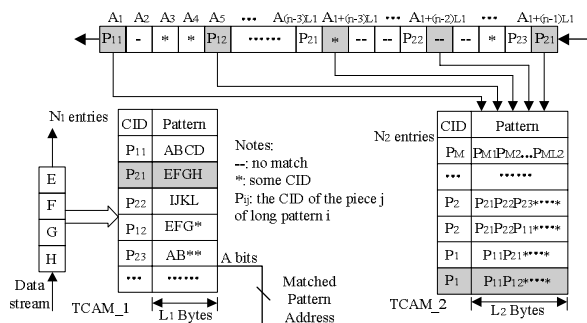


Fig. 1. A sample of cascade TCAMs for long pattern matching problem

than "EFGH". Thus, when "EFGH" is reported, it means both "EFGH" and "EFG\*" are matched, and when "EFG\*" is reported, it means only "EFG\*" but not "EFGH" is matched. However, if the patterns are placed in the other order, they can not be identified. The situation above is called "true-inclusion" in this paper.

Given the total number of the short patterns and pieces of long patterns as N entries, the width of the output index of TCAM\_1 is  $A = INT(\log_2 N)$  bits. (Definition: INT(n) is defined as the smallest integer no less than n in this paper.)

When a match is reported in TCAM<sub>1</sub>, the matching result will be processed like this: on the one hand, the result will be pushed into the FIFO of TCAM<sub>2</sub> with the clock synchronously; on the other hand, the result as an A bits index will be sent into a true value table to check out whether it is an index of short pattern. Only when it returns “true”, the index will be reported.

#### 4.2. Configuration and work flow for the back TCAM (“TCAM<sub>2</sub>” in Fig. 1.):

The data width requirement of TCAM<sub>2</sub> is restricted with following factors: the length of the longest pattern (K bytes), the width of the output index of TCAM<sub>1</sub> (A bits) and the available width modes of particular chips (e.g., 2<sup>n</sup> bytes).

TCAM<sub>2</sub> can be simply implemented with a single TCAM chip. However, there are something different between the FIFO of TCAM<sub>1</sub> and TCAM<sub>2</sub>. Firstly, in every clock cycle, 8 bits network stream data shift into the FIFO of TCAM<sub>1</sub>, while A bits of matching result from TCAM<sub>1</sub> shift into the FIFO of TCAM<sub>2</sub> when a match happens in TCAM<sub>1</sub>. If there is no match happened in TCAM<sub>1</sub>, an A bits of particular promissory invalid data will be fill in the FIFO of TCAM<sub>2</sub> in the clock cycle. Secondly, assume that the longest pattern has been cut into n entries L<sub>1</sub> bytes pieces stored in TCAM<sub>1</sub>, the FIFO of TCAM<sub>2</sub> should be configured as A bits width and (n-1)\* L<sub>1</sub>+1 entries depth. If A<sub>i</sub> is defined to stand for the i<sup>th</sup> item of the FIFO, the input vector of TCAM<sub>2</sub> is merged with n pieces of ordinal items, A<sub>1</sub>, A<sub>1+L<sub>1</sub></sub>, ..., A<sub>1+(n-1)\*L<sub>1</sub></sub>. The indexes of the partial pieces of a long pattern are merged in order and stored in one entry of TCAM<sub>2</sub> as the new pattern for the original long pattern. In Fig. 1., it shows the scene when long pattern P<sub>1</sub>, “ABCDEFGH”, is matched. What’s more, it can be foreseen that long pattern P<sub>2</sub>, “EFGHIJKLAB”, will be matched after (n-3)\* L<sub>1</sub>-1 clock cycles.

Because of the “true-inclusion” situation in TCAM<sub>1</sub>, one long pattern may have several possible combinations of P<sub>ij</sub> which will occupy several entries in TCAM<sub>2</sub>. In Fig. 1., both the “P<sub>11</sub>P<sub>12</sub>” and “P<sub>11</sub>P<sub>21</sub>” entries indicate the long pattern P<sub>1</sub> because of the “true-inclusion” situation between “EFGH” and “EFG\*”.

#### 5. Solution for range matching

The popular method by encoding the existing ranges of ports to match the range fields in header rules has been abandoned because of its large hardware cost and inefficiency when there are lots of different ranges in the set. A more efficient solution, which can report

multiple matching results simultaneously using particular Range Matching Engine (RME), is represented in Fig.2.

The whole system consists of five main parts: TCAM, Index Control Logic, and Range List in Block RAM (BRAM), RME and Multiple Results Mask. Tables in Fig. 3 show the configuration details of TCAM, Index Control Logic and Range List in Block RAM and the structure of the Range Matching Engine.

Table A in Fig. 3 shows a subset of SNORT header rules with 2 pieces of simple header rules and 3 pieces of range header rules. Let each HID<sub>1</sub> in the table be the original index for each rule.

The practical memory organization in TCAM is shown in Table B in Fig. 3. Two points should be noticed here. On the one hand, two original rules (HID<sub>1</sub> = 2 and HID<sub>1</sub> = 5) have the same expression after

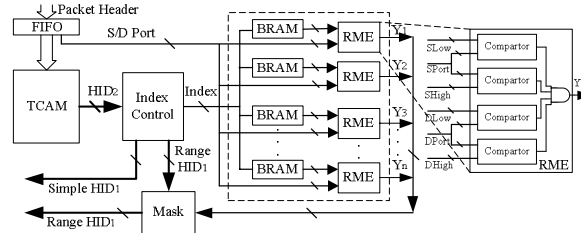


Fig. 2. Structure and diagram flow of header rules matching solution for SNORT

being padded with “do not care” bits. So they are stored as one entry in TCAM with the same new index, HID<sub>2</sub> = 4. On the other hand, similar with the situation in storing signatures in TCAM, these header rules should also be organized in particular order to identify all matching patterns. So patterns have been mapped into new indexes in TCAM, called HID<sub>2</sub>.

Table A.

HID <sub>1</sub>	Source IP	Destination IP	Source Port	Destination Port	Protocol
1	EXTERNAL_NET	HOME_NET	110	23	ICMP
2	EXTERNAL_NET	any	any	any	TCP
3	EXTERNAL_NET	any	[4096, 8000]	80	TCP
4	EXTERNAL_NET	any	[80, 4000]	any	TCP
5	EXTERNAL_NET	any	[80, 65535]	any	TCP

Table B. (in TCAM)

HID <sub>2</sub>	HID <sub>1</sub>	Source IP	Destination IP	Source Port	Destination Port	Protocol
1	1	EXTERNAL_NET	HOME_NET	0x6E	0x17	ICMP
2	3	EXTERNAL_NET	*	0x1***	0x50	TCP
3	4	EXTERNAL_NET	*	0x0***	0x****	TCP
4	2,5	EXTERNAL_NET	*	0x****	0x****	TCP

Table C. Index Control Logic

HID <sub>2</sub>	Index	Range	Simple
1	--	--	1
2	1	3,5	--
3	2	4,5	--
4	3	5	2

Table D. Range List

Index	SL	SH	DL	DH
1	4096	8000	0	65535
	80	65535	0	65535
2	80	4000	0	65535
	80	65535	0	65535
3	80	65535	0	65535

Table D<sub>1</sub>.

Index	SL	SH	DL	DH
1	4096	8000	0	65535
2	80	4000	0	65535
3	80	65535	0	65535

Table D<sub>2</sub>.

Index	SL	SH	DL	DH
1	80	65535	0	65535
2	80	65535	0	65535

Fig. 3. The details of components for an example of the header rules matching solution

There are three fields in the Index Control Logic as the example shown in Table C in Fig. 3. "Index" is used as the address to access the Block RAMs to get the ranges stored in Range List. "Range HID<sub>1</sub>" gives a series of original indexes of header rules, which are possible to be eventually matched when the same HID<sub>2</sub> is reported in TCAM. "Simple HID<sub>1</sub>" shows the original index of the header rule mapped to the HID<sub>2</sub>. For instance, when HID<sub>2</sub> = 4 is reported in TCAM, the legal bytes "2" will be sent out as the "Simple HID<sub>1</sub>" of the matched rule, and "Index = 3" will be sent to the BRAM to get the ports ranges of the header rule whose "Range HID<sub>1</sub>" = 5.

Table D shows the logic structure of the Range List which maintains the ranges of source ports and destination ports for all the header rules. However, it is possible that several "Range HID<sub>1</sub>" may be found in Index Control Logic by the same "Index". Thus, one index in Range List may have several corresponding range couples, which may cost several clock cycles to be read out for compare. So the Range List like Table D can be separated into several small Range Lists in several BRAMs for reading several ranges in parallel. If there are no more than k "Range HID<sub>1</sub>" for one "Index", k sub-tables could make k couples of ports ranges read out in O(1) time in parallel. The i<sup>th</sup> sub-table consists of the i<sup>th</sup> ranges of each "Index". If there is no the i<sup>th</sup> ranges of some "Index", the entry should be filled with the particular promissory invalid data. For instance, the logic structure of Table D can be realized by Table D<sub>1</sub> and Table D<sub>2</sub> in two different BRAMs.

The couple of ports ranges and the ports values of the current packet header are sent to the RME for each BRAM. Each RME consists of four 16 bits comparators, and if the ports values follow the restriction of the ranges, the RME will set the output Y = 1. All the outputs make up a mask vector which will decide which header rules will be finally matched and their "Range HID<sub>1</sub>" will be reported out.

## 6. Analysis of the scheme

### 6.1 Payload content matching engine

The length of the longest pattern, which the payload contents matching engine can handle using cascade TCAMs, is mainly decided by the maximum data width of TCAM and the total entries stored in the front TCAM. The maximum data width of the TCAM is 64 bytes in the current state of arts. Assume that there are N<sub>1</sub> = 4096 entries of pattern in the front TCAM, the cascade TCAM system can handle the signatures with the length up to 2688 bytes. If necessary, another TCAM can be cascaded to the system, so that the

system can handle the signatures with the total length of hundreds of thousands of bytes. The several possible configurations of cascade TCAM system for current SNORT signature set have been given in Table 2. Different choices can be made for system requirements on different goals. TCAM<sub>2</sub> can be implemented by TCAM IP cores in FPGA to save the back TCAM chip when the width and depth requirements are low enough. All the numbers shown in the "Depth" field of TCAM<sub>1</sub> in Table 2 are the sum of amount of the short patterns and pieces of long patterns. The overlap situation is not considered. So the practical numbers will no more than these in Table 2. But those of TCAM<sub>2</sub> are just the amounts of long patterns (M), while the "true-inclusion" situation is not considered. The practical numbers will no less than these in Table 2.

### 6.2 Header rules matching engine

Functionally, the header rules matching engine for SNORT set using the solution represented in Section 5 can report multiple matching results simultaneously.

As to the performance, such a packet classification system do not need the expansion of TCAM entries for range matching, so the TCAM can be configured to 16 bytes width and 300 entries depth to handle current SNORT header rules set. By using the state-of-arts TCAM, the device can finish the searching task for 128k entries header patterns within 4 ns. Considering the worst case, assume that all the 300 header rules have the range restrictions for both ports and each "Index" can be relevant with 20 pairs of ranges at most, then 20 BRAM will be needed, 2400 bytes for each BRAM at least. Totally 48000 bytes BRAM will

**Table 2. Configurations of cascade TCAMs system for SNORT signature matching**

TCAM <sub>1</sub>			TCAM <sub>2</sub>			System Thruput (Gbps)
Width (bytes)	Depth (entry)	Chip Freq. (MHz)	Width (bytes)	Depth (entry)	Chip Freq. (MHz)	
64	2085	66.5	3	3	266	0.532
32	2340	133	6	253	266	1.064
<b>16</b>	<b>2976</b>	<b>266</b>	<b>12</b>	<b>610</b>	<b>266</b>	<b>2.128</b>
8	4904	266	26	1418	133	1.064
4	8694	133	55	1770	66.5	0.532

be needed in the worst case. However, the latency of reading operation for BRAM in Xilinx Virtex2 pro 30 is only 2.5 ns. 20 sets of RME in the same FPGA only cost about 1700 logic cells and can reach 250MHz frequency.

To sum up, if the five parts of the system work in pipeline mode, it can handle the packet classification task for SNORT header rules at the rate of 250 million packets per second in theory, which means 375Gbps in

normal frame length and 16Gbps even when the packets size are all only 64 bytes.

## 7. Conclusion

Addressing the drawbacks of TCAM in long pattern matching and range pattern matching, this paper represented two novel and efficient corresponding solutions respectively. On the one hand, the payload contents matching engine based on cascade TCAMs can handle large signature set made up of patterns with thousands or even hundreds of thousands of bytes length at deterministic high data rates. For current SNORT signature set, it can perform multi-pattern matching at rates of 2Gbps (one single set of engine). On the other hand, the header rules matching engine based on range pattern matching structure proposed above can perform the packet classification task for current SNORT header rules at the rate of up to 250 million packets per second in theory.

In fact, the technology of TCAM is developing fast now. The TCAM macro with 144 bits width and 512 entries, which can perform 800 mega look-ups per second, has been proven in silicon and available in TSMC CL013LV/LVOD process [9]. To building security platforms for very high speed network on TCAMs must be a hot research topic in the future.

## Acknowledgments

The authors are devoutly thankful for the supports of the Hi-Tech Research and Development Program of China (863 Program), Project 2001AA413910.

## References

- [1] Martin Roesch, "SNORT - lightweight intrusion detection for networks," in LISA '99: USENIX 13th Systems Administration Conference on System administration, Seattle, Washington, Nov. 1999, pp. 229–238.
- [2] Lambert Schaelicke, Thomas Slabach, Branden Moore et al, "Characterizing the performance of network intrusion detection sensors," Proceedings of the Sixth International Symposium on Recent Advances in Intrusion Detection (RAID 2003), Berlin–Heidelberg–New York, September 2003, Lecture Notes in Computer Science, Springer-Verlag.
- [3] C.J. Coit, S. Stanford, and J. McAlerney, "Towards Faster String Matching for Intrusion Detection or Exceeding the Speed of Snort", DARPA Information Survivability Conference and Exposition, (DISCEX II'01), 2001.
- [4] Ioannis Sourdis and Dionisios Pnevmatikatos, "Pre-decoded CAMs for efficient and high-speed NIDS pattern matching" in IEEE Symposium on Field-Programmable Custom Computing Machines, (FCCM), Napa, CA, Apr. 2004.
- [5] Christopher R. Clark and David E. Schimmel, "Scalable multi-pattern matching on high-speed networks," IEEE Symposium on Field-Programmable Custom Computing Machines, (FCCM), Napa, CA, Apr. 2004.
- [6] Y. Cho and W. Mangione-Smith, "Deep packet filter with dedicated logic and read only memories," in IEEE Symposium on Field-Programmable Custom Computing Machines, (FCCM), Napa, CA, Apr. 2004.
- [7] Michael E. Attig, Sarang Dharmapurikar, and John Lockwood, "Implementation results of Bloom filters for string matching," in IEEE Symposium on Field-Programmable Custom Computing Machines, (FCCM), Napa, CA, Apr. 2004, pp. 322–323.
- [8] Fang Yu, Randy H. Katz, et al., "Gigabit Rate Packet Pattern-Matching Using TCAM," in IEEE International Conference on Network Protocols, (ICNP), Berlin, Germany, Oct. 2004, pp.174-183
- [9] Analog Bits Inc., "High Speed TCAM Datasheet", 2004