FPGA based High speed and low area cost pattern matching

Jian Huang, Zongkai Yang, Xu Du, and Wei Liu
Department of Electronic and Information Engineering, Huazhong University of Science and Technology
hjshark@126.com, zkyang@public.wh.hb.cn, duxu@263.net, wliu@public.wh.hb.cn

Abstract—Intrusion detection and prevention system have to define more and more patterns to identify the diversification intrusions. Pattern matching, the main part of almost every modern intrusion detection system, should provide exceptionally high performance and ability of reconfiguration. FPGA based pattern matching sub-system becomes a popular solution for modern intrusion detection system. But there is still significant space to improve the FPGA resource efficiency. In this paper, we present a novel pattern matching implementation using the Half Byte Comparators (HBC). HBC based pattern matching approach can increase the area efficiency. But the operating frequency will be a little decrease. We also explored some methods to improve the operating frequency in this paper. The result shows for matching more than 22,000 characters (All the rules in SNORT v2.0) our implementation achieving an area efficiency of more than 3.13 matched characters per logic cell, achieving an operating frequency of about 325 MHz (2.6Gbps) on a Virtex-II pro device. When using quad parallelism to increase the matching throughput, the area efficiency of a logic cell is decrease to 0.71 characters for a throughput of almost 8.5 Gbps.

Index Terms—FPGA, Half-byte Comparator, Intrusion Detection System, LUT, Register, Combination Logic, Pattern Matching, Rule, SNORT

I. Introduction

Network security becomes a hot topic nowadays. Methods commonly used to protect against network attacks include firewalls with packet filter to filter out obviously dangerous packets, and Intrusion Detection Systems (IDS) which use much more sophisticated rules and pattern matching to distinguish potential dangerous packets. But these techniques require huge computing powers of network security devices. The traditional software solution is not competent for the high speed networks nowadays [14]. Hardware based solution can meet the performance requirements of the today and tomorrow's networks. The key module of the hardware based network security device is pattern matching.

The signature of an attack may exist at any position of data packets in network traffic. In order to identify if there is any of the predefined patterns existing in the target packet, pattern matching module should inspect the packet byte by byte. In general, the input of pattern matching system is one byte per clock period. In order to improve the throughput of the pattern matching module, the input will be parallel N-bytes per clock period. The output of string matching system are matching signal and pattern index. The matching signal indicates whether there is predefined pattern matched. The pattern index indicates the existence of predefined pattern in the target data packets. The patterns defined by the SNORT [10], a well known open source software based IDS, are often used in all kinds of IDS. It defines thousands of patterns in its anti-attack rules. In order to check input packets in wire speed, the pattern matching module should compare the packet data with all the predefined patterns synchronously when the packet passes by. The parallel compare is the most important and complex part in hardware based pattern matching system.

Hardware based pattern matching system has the advantages of high speed and parallel processing [6]. It can provide high throughput at multi-giga bits per second. But such system should consider two issues: how to reduce the hardware resource consumption and how to have the reconfiguration ability. FPGA based pattern matching system can deal with the second issue very well, which make it widely used in the nowadays IDS. However, the resource in FPGA is limited. With the diversifying trend of network attack methods, more and more SNORT patters are defined. The latest SNORT [10] version (v2.32) defines almost 5,600 patterns (more than 57,000 characters). It is difficult to implement those patterns in just a single FPGA chip. Thus, improve the area efficiency of FPGA resource is necessary.

In this paper we advocate using HBCs in FPGA based pattern matching module. Because of the share of the comparing results, our pattern matching implementation can improve the area efficiency in FPGA significantly. Thousands of predefined matching patterns can be implemented in a single FPGA chip. Combined with some timing improvement methods, our approach can operate at a very high speed which can meet the performance requirement of the giga bits Ethernet, OC-48 (2.5Gbps), even if the OC-192 (10Gbps) networks.

The rest of this paper is organized as following: Section II reviews the previous related work; Section III introduces the architecture of HBC based pattern matching module; Section IV proposes some methods to improve the throughput of pattern matching module; and Section V present the evaluation results of the pattern matching module implementation; Finally Section VI concludes this paper.

II. Related works

FPGA based pattern matching can provide high speed and ability of reconfiguration. In order to deal with the area efficiency issue, many methods are investigated in our previous work:

- In regular expression matching [7, 12], the authors proposed to use Non-deterministic Finite Automaton in matching regular expressions and implemented it on a FPGA. They also provided a simple, fast algorithm that can quickly construct the NFA for the given regular expression.

- In [4], a KMP (a well known software algorithm proposed in 1977[17], it was proposed by D.E. Knuth, J. Morris, and V.R. Pratt, shorted in KMP) algorithm based on FPGA implementation was proposed. It can support rule reconfiguration by modifying the patterns in the distributed RAM and block RAM in FPGA. But it only achieves low throughput and low FPGA resource efficiency.
- In [5], a novel hashing mechanism using the method of Bloom filter was discussed. The authors implemented the pattern matching block based on hashing-table lookup. This approach only uses a moderate amount of logic and external or internal memory. It is an efficient method to compare thousands of strings in a single pass and the rules can be added or modified in the memory. However, its implementation relies on the FPGAs with multi-ports block RAMs, which is very expensive. And because of using complex RAM and hash algorithm, the system speed was limited.
- In [14], a deep packet filter using dedicated logic and read only memory was introduced. The author proposed two effective methods to design dynamic pattern search engines in FPGA. The first method uses RDL (Reconfigurable Discrete Logic) to detect dynamic patterns in data stream. The second method uses built-in memory in the FPGA and XOR based comparators.
- In [1,2,3,11], a Content Addressable Memory (CAM) based software/hardware IDS was proposed. CAM is used to match against possible attacks contained in a packet. These CAM based pattern matching approaches have very high system throughputs. While have lower area efficiency.

The above techniques have the following shortcomings: 1) byte comparator does not fit the 4-input LookUp Table (LUT) in FPGA [16]. 2) Large numbers of fan out of input signals and comparator outputs will cause the decrease of system operating frequency. 3) The amount of comparators will increase with the number of pattern of system. 4) The last disadvantage is the waste of FPGA source. The two main kinds of resources in FPGA are Combination Logic and Register. The above approaches focus on combination logic and their excessively using of Combination Logic wastes the register resource.

III. Architecture of Pattern Matching

The main idea of this paper is to use half-byte comparators (HBC) in the pattern matching system. HBCs are simply 4 to 1 decoders. If the input 4 bits match the value configured in the HBC, the output of HBC will be asserted high. A HBC just fits an FPGA LUT [16]. Every character needs two Half-byte comparators, for higher 4bits and lower 4bits respectively. In our design, the same higher/lower 4 bits comparators are fully shared.

Taking "ABCA" as an example pattern, we illustrate the system architecture in Figure 1. The ASCII code of "ABCA" in HEX is |41 42 43 41|, as a result, 8 comparators are needed in normal matching implementation. Since the four higher 4 bits comparators all compare to 0x4, it can be shared. In the same idea, the comparator for 0x1 of the lower 4 bits can also be shared. By this way, the pattern "ABCA" only needs 4 HBCs. As shown in Figure1, the module has one byte input per clock period (cycle). If we assume the first character of string "ABCA" arrived at cycle *t*, then the higher HBC for 0x4 and lower HBC for 0x1 will be matched. Because there are 3 delay registers at the output side of HBCs for comparing 0x4 and 0x1, these two matched signal will be sent to the AND gate at cycle *t*+3. And at cycle *t*+1, character "B" arrived, its match signal will also be sent to the AND gate at cycle *t*+3. The rest characters will be processed in the same way. So we can get the pattern matching signal at cycle *t*+3. These delayed half-bytes matching signals will be AND-ed by an 8 inputs AND gate. The output of AND gate is the final pattern matching signal.

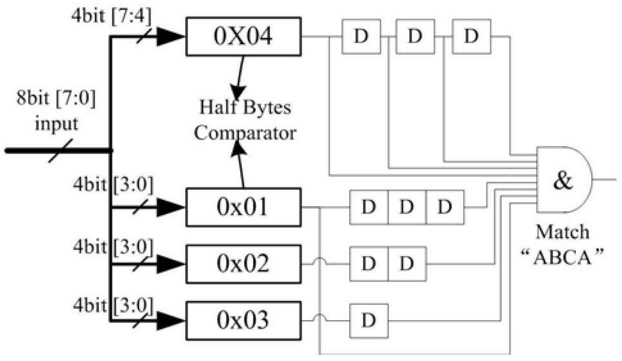
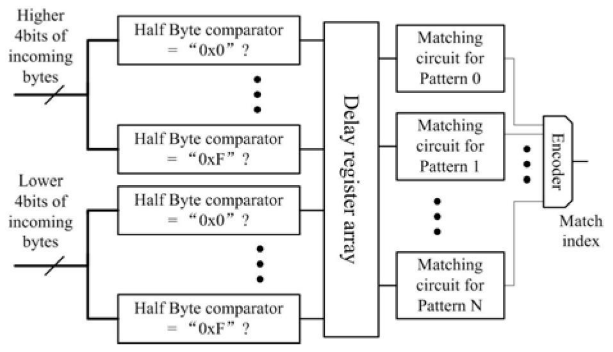


Figure 1, HBC architecture for pattern "ABCA" matching. The output of comparator for 0x04 is shared in comparing the higher 4 bits for all the 4 characters. The output of comparator for 0x01 is shared in comparing lower 4 bits for the first and last character "A".

The patterns of SNORT system are made up of ASCII characters and bytes in HEX. There are total 256 kinds of variety for a byte. After dividing a character into two half bytes, every half byte has 16 kinds of variety. Then there are only 32 kinds of variety in total. So, pattern matching system using HBCs can set total 32 HBCs to compare the incoming bytes simultaneously. The outputs of these comparators connect to a well organized delay register array. The delay register array is used to generate right character input sequence for every SNORT pattern. Dedicated pattern matching circuits are designed for every pattern. Necessary delayed Half-byte matching signal is send to every matching circuit. For example, matching circuit for pattern "ABCD" needs 8 delayed Half-byte matching signals: matching signals of 0x4 and 0x1 (delayed 3 cycles) for "A", 0x4 and 0x2 (delayed 2 cycles) for "B", 0x4 and 0x3 (delayed 1 cycle) for "C", 0x4 and 0x4 (delayed 0 cycle) for "D". Matching circuit generates a final signal which tells the encoder if the pattern did match. Figure 2 shows the implementation for all SNORT patterns.



Higher 4bits of incoming bytes
Lower 4bits of incoming bytes
Half Byte comparator = "0x0" ?
Half Byte comparator = "0xF" ?
Delay register array
Matching circuit for Pattern 0
Matching circuit for Pattern 1
Matching circuit for Pattern N
Encoder
Match index

Figure 2, Using HBCs to implement all the SNORT Rules, There are total 32 HBCs, the output of delay register array is shared. Every pattern has a dedicated matching circuit. The matching circuit connects the necessary delayed compare results as inputs.

As shown in Figure2, HBCs and matching circuit use the LUT and routing resources in FPGA. The delay register array uses the used filp-flop resource in FPGA. Because the outputs of delay register array are shared, the used amount of LUTs is larger than the used flip-flop resource. We found the used amount of LUT resource is related to the amount of target characters. It is doable to partition the SNORT rules into small groups. This partition can reduce the complexity of the total matching circuit and improve the system operating frequency.

IV. Methods to improve Performance

A. M bytes Parallel matching

In order to improve the performance of pattern matching system, we designed M bytes in parallel pattern matching module. We can widen the input bus by a parameter of M providing M copies of HBC pattern matching group and the corresponding matching circuits.

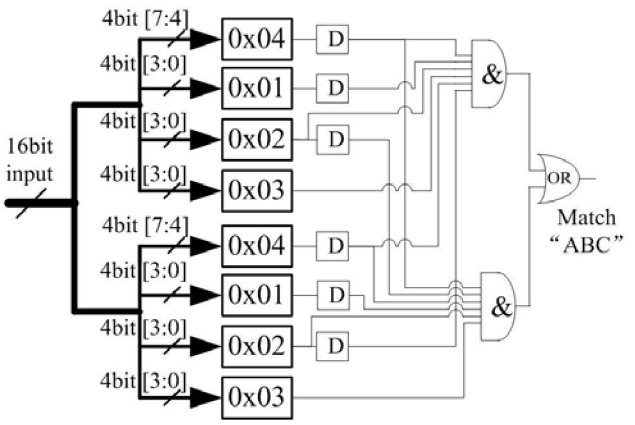


Figure 3, Doubled HBC groups process 2 input bytes per cycle. The start character of the predefined matching pattern may occur at the higher byte or lower byte of the input parallel two bytes

Figure 3 shows a pattern module of M = 2. There are two conditions for the alignment of pattern exists in the target packet. The start character of the predefined pattern may occur at the higher byte or lower bytes of the input parallel two bytes. We must design two pattern matching circuits for the two conditions. Thus, the resource in FPGA is doubled.

If the start character of the predefined matching pattern occurs at the higher byte, output of the upper AND gate in figure 3 will be asserted. If it occurs at the lower byte, output of the bottom AND gate in figure 3 will be asserted high. Ether of the two conditions will cause the OR gate outputs asserted high. For any value of M, this scheme can be used.

B. Decreasing the numbers of fan out

Because of the output of delay register array shared, the fan out of some register will be as large as hard to implement in FPGA with high system frequency. In order to decrease the numbers of fan out of these registers, we add some registers to share in the fan out. Figure 4 illustrates the sharing method to decrease the numbers of fan out.

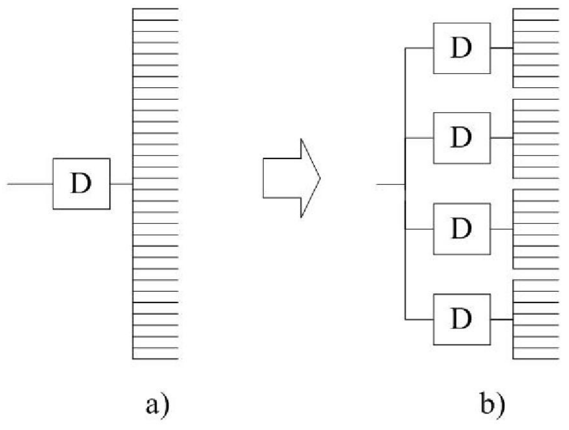


Figure 4, Using registers to solve the large numbers of fan out problem. A register with 32 fan outs in a) can be replaced by 4 registers with 8 fan outs in b).

C. Decreasing the combination logic delay

Propagation and gate delay of combination logic can be ended by registers. The combination logic in Xinlinx [16] makes up of LUTs and routing resources. The inputs of AND gates larger than 4 bits will cause the stacking of LUTs, which will increase the propagation delay. In order to decrease the propagation delay, registers and small AND gates are used. Figure 5 shows the method to decrease the propagation delay.

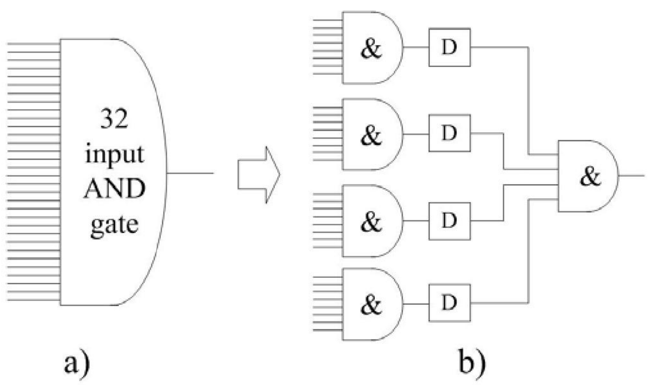


Figure 5, Using registers to the huge combination logic. 32 inputs AND gate in a) can be divided into two groups of small AND gate (8 inputs and 4 inputs) which separated by registers in b).

V. Evaluations

There are two important metrics generally used to evaluate the pattern matching modules: performance and area cost. Performance can be presented by the system operation frequency and throughputs. And area efficiency can be presented by the implemented characters per logic cell. In order to get these two metrics, the implementation of pattern matching system should be done. Considering the mapping from patterns to RTL code is simple in our design, we developed a program that can translate the specified patterns into synthesizable Verilog HDL. We implemented the SNORT rule set v2.0 [9] with M=1 and M=4 on a Vertex2 pro 30 FPGA. The speed grade of FPGA is -5. The synthesis tool is Xilinx ISE version 7.1.

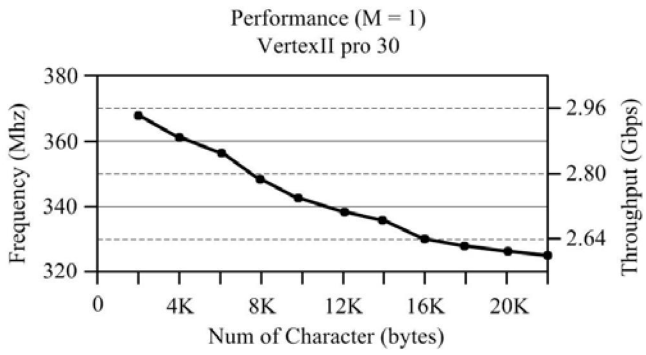


Figure 6, performance analysis of HBC based pattern matching implementation. The parallelism factor M = 1. The max system clock is 325 Mhz when implement all the SNORT v2.0 rules (more than 22,000 characters). And the max throughput is 2.6Gbps.

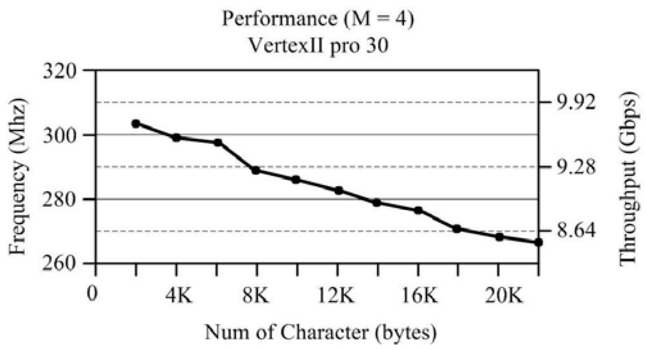


Figure7, performance analysis of HBC based pattern matching implementation. The parallelism parameter M = 4. The system clock can be 268 Mhz when implement all the SNORT v2.0 rules. And the max throughput is 8.576Gbps.

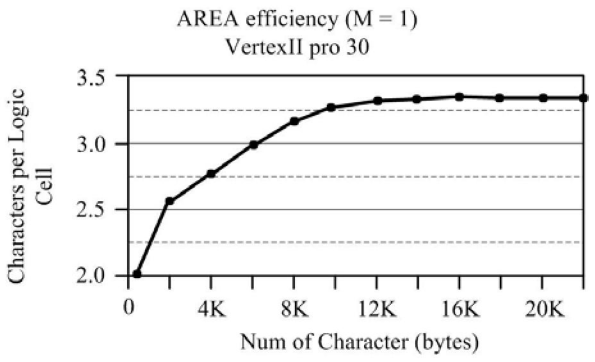


Figure 8, area efficiency analysis of HBC based pattern matching implementation. The parallelism parameter M = 1. With the increasing of matched characters, the matched characters of a logic cell will increase. The Max area efficiency will in crease to 3.13 matched characters per logic cell when implementing all the SNORT v2.0 rules.

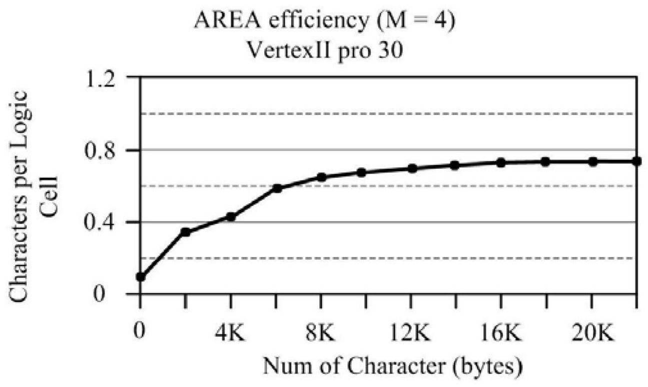


Figure 9, area efficiency analysis of HBC based pattern matching implementation. The parallelism factor M = 4. The max area efficiency is 0.71matched characters per logic cell.

Figure 6 shows the evaluation result of performance when the parallel parameter equals 1. And figure 7 shows the evaluation result of performance when the parallel parameter equals 4. Figure 8 shows the evaluation result of area efficiency when the parallel parameter equals 1. And figure 9 shows the evaluation result of area efficiency when the parallel parameter equals 4. The operating frequency is decrease when implemented character increasing but the area efficiency is increased when implemented character increasing.

VI Conclusions

In this paper, we present a novel pattern matching approach using the HBCs. With our approach, tens of thousands of characters can be implemented in a single FPGA chip. The following methods are used in our approach. First, we reduce the area cost of character using (i) 32 HBCs to buildup all the possible ASCII character and (ii) register array and combination logic to implement all possible character combination for SNORT signatures. Second we achieve high operating frequencies by (iii) using parallel HBC group for processing M bytes parallel input and (iv) using state-of-the-art pipelines for faster circuits.

In a Xilinx Vertex II pro 30 FPGA, we implemented the entire rule set for SNORT version 2.0 (more than 22,000 characters for 2,000 patterns). The pattern matching module totally used 7,000 logical cells. The area efficiency is 3.13 characters per Logic Cell. And the system operation frequency is about 325MHz (about 2.6Gbps). When using quad parallelism to increase the throughput, the throughput can be increased to almost 8.576 Gbps (268 MHz) with acceptable decrease of the area efficiency to 0.71 characters per Logic Cell. TABLE I shows the compare result with other pattern matching approaches. It shows our approach has area efficient advantage. The operating frequency is a little lower than the pre-decode CAM which is the fastest approach.

TABLE I

Result of performance and efficiency comparison with other pattern matching approaches

Description	Device	Frequency (Mhz)	Throughput (Gbps)	Implemented Bytes	Chars /LC
Huang, el Half-byte Comparators	Virtex II pro	268	8.576	22,138	0.71
	Virtex II pro	325	2.60	22,138	3.13
Sourdis, el Pre-decoded CAMs [2]	Virtex2	385	9.708	18,031	0.28
Bu Long, el, CAMs[1]	Virtex II Pro	281	2.251	~560	0.33
Geir Nilsen, el, Variable Word-Width CAM[9]	Virtex II Pro	100	0.8	3,601	0.20
Zachary K, el, KMP algorithm in FPGA [4]	Virtex II Pro	285	2.4	-	-
Wash U, el. Bloom filter using hash[5]	XCV2000E	81	2.592	1,434	0.13

References

[1] Bu Long, J.A. Chandy. FPGA based network intrusion detection using content addressable memories. Field-Programmable Custom Computing Machines (FCCM 2004), Pages 316-317, Apr 2004.

[2] I. Sourdis, D. Pnevmatikatos. Pre-decoded CAMs for efficient and high-speed NIDS pattern matching. Field-Programmable Custom Computing Machines(FCCM 2004), Pages 258-267, Apr 2004.

[3] G. Nilsen, J. Torresen, O. Sorasen. A variable word-width content addressable memory for fast string matching. Proceedings of Norchip Conference 2004, Pages 214-217, Nov 2004.

[4] Zachary K. Baker, Viktor K. Prasanna. Time and Area Efficient Pattern Matching on FPGAs. Field-Programmable Custom Computing Machines (FCCM 2004), Pages 125-134, Feb 2004.

[5] Sarang Dharmapurikar, Praveen Krishnamurthy, T.S. Sproull, J.W. Lockwood. Deep packet inspection using parallel bloom filters. Micro, IEEE. Volume 24, Issue 1, Pages 52-61, Jan. 2004.

[6] M. Fisk and G. Varghese. An analysis of fast string matching applied to content-based forwarding and intrusion detection. In Techical Report CS2001-0670 (updated version), University of California - San Diego, 2002.

[7] R. Sidhu, V.K. Prasanna. Fast Regular Expression Matching Using FPGAs. Field-Programmable Custom Computing Machines (FCCM 01), Pages 227-238, 2001.

[8] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood. Implementation of a Deep Packet Inspection Circuit using Parallel Bloom Filters in Reconfigurable Hardware. In Proceedings of HOTi03, 2003.

[9] Geir Nilsen, Jim Torresen and Oddvar Søråsen. A Variable Word-Width Content Addressable Memory for Fast String Matching. In Norchip, 2004.

[10] Sourcefire. Snort: The Open Source Network Intrusion Detection System. http://www.snort.org, 2005.

[11] I. Sourdis and D. Pnevmatikatos. Fast, Large-Scale String Match for a 10Gbps FPGA-Based Network Intrusion Detection System. In Proceedings of FPL2003, 2003.

[12] R. Franklin, D. Carver, and B. Hutchings. Assisting network intrusion detection with reconfigurable hardware. In IEEE Symposium on Field-Programmable Custom Computing Machines, April 2002.

[13] M. Gokhale, D. Dubois, A. Dubois, M. Boorman, S. Poole, and V. Hogsett. Granidt: Towards gigabit rate network intrusion detection technology. In Proceedings of 12th International Conference on Field Programmable Logic and Applications, France, 2002.

[14] Young H. Cho and William H. Mangione-Smith. Deep Packet Filter with Dedicated Logic and Read Only Memories, Field-Programmable Custom Computing Machines (FCCM 2004).

[15] N. Desai. Increasing performance in high speed NIDS. In www.linuxsecurity.com, March 15 2002.

[16] Xilinx. Virtex-II Platform FPGAs: Detailed description. http://direct.xilinx.com/bvdocs/publications/ds083.pdf, October 2004.

[17] D.E. Knuth, J. Morris, and V.R. Pratt. Fast Pattern Matching in Strings. In SIAM Journal on Computing, 1977.