Military Technical College
Kobry El-Kobbah,
Cairo, Egypt

6th International Conference
on Electrical Engineering
ICEENG 2008

A reconfigurable multi-byte regular-expression matching architecture

By

Tamer Farouk Badran* Hany H. Ahmad** Mohamad Abdel-Gawad ***

Abstract:

String/Regular-Expression Matching is widely used in different applications. Our work is concerned with high-throughput regular-expression matching in the context of Intrusion Detection Systems as it is the most computationally intensive part of the operation. The results, however, should be equally applicable to other domains that require fast regular-expression matching. The major contribution of this paper is a reconfigurable architecture that performs regular-expression matching on a multi-byte per clock cycle basis. We are able to explore the system performance for different byte-processing rates – from 4 to 64 – by automating the VHDL-generation process and implementing the resulting circuits on a general-purpose FPGA. Theoretical expressions for resource usage (cost) as a function of byte-rate and pattern-length are also presented.

Keywords:

String Matching, Regular-Expression Matching, Intrusion Detection, Reconfigurable Architecture

* Teaching Assistant, Assiut University, Assiut, Egypt
** Lecturer, Assiut University, Assiut, Egypt
*** Professor, Assiut University, Assiut, Egypt

1. Introduction:

Today's crucial information networks are vulnerable to fast moving attacks by internet worms and computer viruses. These attacks have infected computers globally, clogged large computer networks, and degraded corporate productivity; so the need for Intrusion Detection Systems (IDSes) is urgent. The ever-increasing data rates of current and future networking standards dictate the need for high throughput matching. For instance, an already existing standard – the OC768 – requires a data rate in excess of 40Gbps. Also, in July 2007, the IEEE 802.3 Higher Speed Study Group (HSSG) presented a project for a new IEEE 802.3ba standard which defines both 40Gbps and 100Gbps data rates. To cope with such high data rates, *dedicated hardware* solutions are attractive since they can naturally exploit the inherent parallelism of the problem. In the particular case of regular-expression (regex) matching in IDSes, *reconfigurability* is also required to accommodate the dynamically changing rule-set of the IDS. So FPGAs are a suitable platform to implement such systems.

The remainder of this paper is organized as follows: in Section 2 we discuss related work, in Sections 3 and 4 we present our work and the results, and finally we summarize the outcome in Section 6.

2. Related Work:

Sidhu and Prasanna [1] presented a detailed structure of the comparators and the circuits to support all the metacharactes found in regex. In their work, regex is converted to NFAs and then implemented on FPGA. The matching process is performed on a single-byte per clock cycle basis. They also automated the generation of the architecture and its placement and routing on their Self-Reconfigurable Gate Array (SRGA).

Another approach of regex matching presented by Hutchings et. al. is to create a JHDL-based module generator [2] capable of handling regex operators, based on standard regex syntax. These circuits are based on the NFA implementation presented in [1] – matching process is performed on a single-byte per clock cycle basis also – and once created they can be debugged and verified with the JHDL simulator and design browser. JHDL emits EDIF net-lists that can be passed to Xilinx place and route software for bit-stream generation. JHDL provides run-time support for debugging the running hardware in the context of the original design using the same GUI as the JHDL simulator.

In [3] Cho et. al. presented an architecture that compares the input packets against the desired pattern at a rate of 4 bytes per clock and took into consideration all the 4 offsets of the pattern to increase the throughput, but with no support for regex metachatacters.

Clark et. al. [4] increased the rate of [3] to take 4, 8, and 16 bytes per clock on Virtex-2 8000 and 16, 32, and 64 bytes per clock on Virtex-2 Pro 125 and investigated the throughput, number of characters, and the resource utilization.

3. Our Work

Based on our literature review reported in [5], we started the exploration of high-throughput sting matching for IDSes by extending the single-byte architecture of Sidhu and Prasanna [1] to handle multi-byte exact string-matching case while taking care of possible pattern offsets in a uniform manner. The result of the first step is the architecture for which a 4-byte (per clock cycle) example is given in **Figure (1)**[1].

Figure (1): Structure of 4-bytes per clock matching circuit for the pattern "abcdef"

[1] It was later brought to our attention that the same architecture had already been reported in [3].

The next step was to extend the architecture to support the metacharacters used in regex ('?'–zero or one times, '*'–zero or more times, '+'–one or more times, '.'–match anything). A 4-byte instance of the new architecture, for matching the regex "a(bcde)$^+$f", is shown in **Figure (2)**. Operation of the circuit when attempting to match the input string "sebcdabcdebcdef" is illustreated in **Table (1)**.

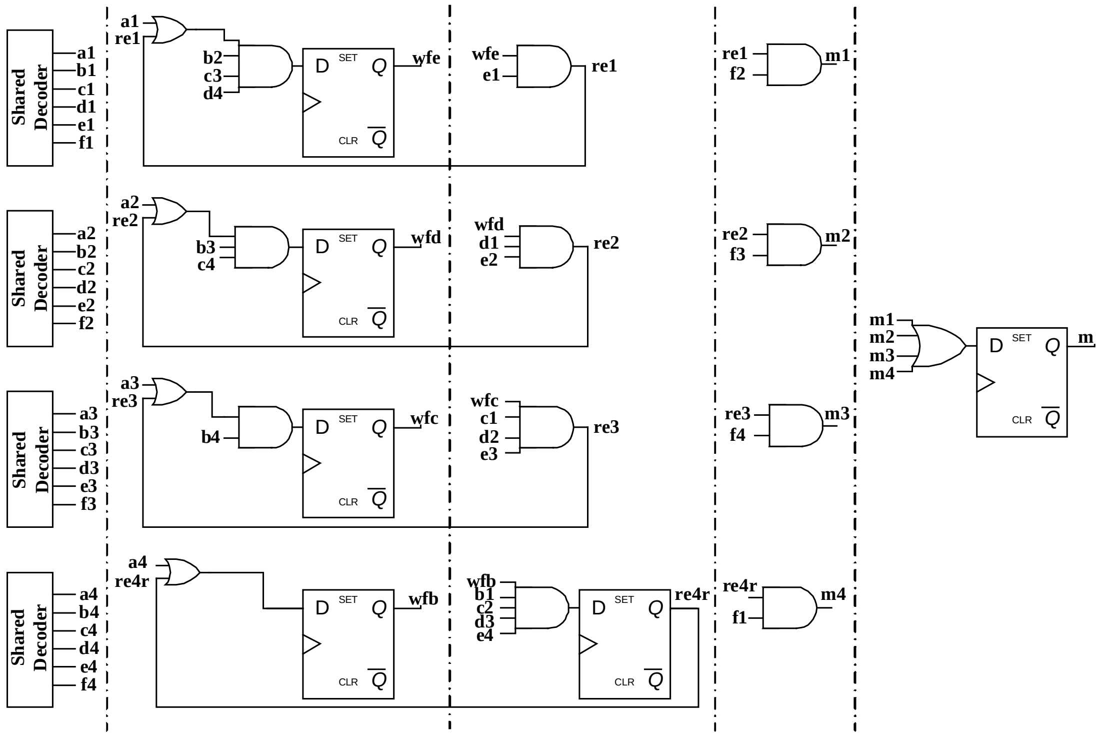

Figure (2): Structure of 4-bytes per clock matching circuit for the pattern "a(bcde)$^+$f"

Table (1): Example of regex matching

Clock Cycle	Active Signals
#1 – "sebc"	s_1 , e_2 , b_3 , c_4 è nothing happens
#2 – "dabc"	d_1 , a_2 , b_3 , c_4 è the waiting for 'd' (wfd) signal will go high
#3 – "debc"	d_1 , e_2 , b_3 , c_4 è re_2 goes high, then ANDed with b_3,c_4 so wfd goes high
#4 – "def"	d_1 , e_2 , f_3 è re_2 goes high, then ANDed with f_3 so m_2 and m go high

The third element of the investigation was to study the FPGA resource consumption as a function of the number of bytes processed per clock cycle (L_J) as well as the length of the target pattern (L_P), are listed in **Table (2)**. To explore the design-space easily and flexibly, a program that takes as input the target regex and the desired bytes-per-clock rate, and generates the corresponding Structural VHDL module was developed. The resulting VHDL was then used to implement the system on target FPGAs using vendor-supplied synthesis tools.

Table (2): Resources consumption as a function of L_J and L_P

Resources	1 byte/clock	L_J byte per clock
Shared Decoders	1	L_J
Flip-Flops	L_P	L_P
2-input OR Gates*	1	L_J
AND Gates	L_p with 2 inputs	(L_J-1) AND Gates: Gate no. x has min (x+1,L_P) inputs (L_P-1) AND Gates: Gate no. y has min (y+1,L_J+1) inputs

* Needed only if the pattern contain regex metacharacters

4. Implementation Results:

The rates of 4, 8, 16, 32, and 64 bytes per clock cycle were tested on three popular FPGA platforms[2], and for each platform, the effect of increasing the number of bytes processed per clock cycle on clock rate, throughput, and resource usage was observed. **Table (3)** lists implementation results of a synthetic rule-set comparable to the complete Snort rule-set (which comprises 1400 rules containing from 2 to 107 characters each), but with no meta-characters included in the rules (i.e. exact pattern matching is assumed). The rule-set statistics used here are a reproduced version of those presented by Ioannis Sourdis in [6]. Clock frequency is rounded down to the nearest 5MHz and Throughput is rounded down to the nearest 0.5 Gbps. This data is included as a baseline for comparison with previous work.

Table (3): Results for complete rule-set assumed to be exact-patterns

Platform	Bytes per cycle	Clock-MHz	Throughput-Gbps	Resource Usage
Spartan-3E	1	165	1	72% of XC3S1600E
	4	150	4.5	87% of XC3S1600E
	8	135	8.5	185% of XC3S1600E (2Chips)
	16	135	17	467% of XC3S1600E (5Chips)
	32	135	34	1073% of XC3S1600E (11Chips)
	64	130	66.5	3861% of XC3S1600E (39Chips)

[2] Xilinx FPGA platforms – Spartan-3E Speed Grade -4, Virtex-2Pro Speed Grade -7, and Virtex-5 Speed Grade -3.

Virtex-2Pro	1	275	2	32% of XCV2P70
	4	255	8	39% of XCV2P70
	8	235	15	80% of XCV2P70
	16	215	27	189% of XCV2P70 (2 Chips)
	32	210	53.5	381% of XCV2P70 (4 Chips)
	64	200	102	880% of XCV2P70 (9 Chips)
Virtex-5	1	425	3	31% of XC5VLX110
	4	415	13	37% of XC5VLX110
	8	355	22.5	64% of XC5VLX110
	16	335	42.5	113% of XC5VLX110 (2 Chips)
	32	280	71.5	236% of XC5VLX110 (3 Chips)
	64	280	143	531% of XC5VLX110 (6 Chips)

With the assumption that all the rules have regex metacharacters the implementation results changed to be as listed in **Table (4)**. If the actual Snort rule-set were implemented, the results will be expected to lie somewhere between those of **Table (3)** and **Table (4)**.

Table (4): Results for complete rule-set assumed to be regular-expressions

Platform	Bytes per cycle	Clock-MHz	Throughput -Gbps	Resource Usage
Spartan-3E	1	115	1	72% of XC3S1600E
	4	110	3.5	99% of XC3S1600E
	8	105	6.5	222% of XC3S1600E (3Chips)
	16	100	12.5	561% of XC3S1600E (6Chips)
	32	100	25.5	1302%of XC3S1600E (14Chips)
	64	100	51	4720%of XC3S1600E (48Chips)

Virtex-2Pro	1	205	1.5	32% of XCV2P70
	4	200	6	45% of XCV2P70
	8	180	11.5	97% of XCV2P70
	16	165	21	227% of XCV2P70 (3 Chips)
	32	160	40.5	463% of XCV2P70 (5 Chips)
	64	155	79	1076% of XCV2P70 (11 Chips)
Virtex-5	1	260	2	31% of XC5VLX110
	4	260	8	43% of XC5VLX110
	8	230	14.5	81% of XC5VLX110
	16	225	28.5	149% of XC5VLX110 (2Chips)
	32	215	55	313% of XC5VLX110 (4Chips)
	64	215	110	709% of XC5VLX110 (8Chips)

Implementation results show that as the byte-rate increases, the critical-path delay increases at a much slower rate, leading to considerable throughput gains. For example, using a Vertix-5 FPGA, a throughput in excess of 110 Gbps is achievable when processing 64 bytes in parallel, compared to 2 Gbps for the single-byte case; a remarkable speedup factor of 55. As expected, however, this throughput gain is only achievable at the expense of increased resource usage. In the previous example, for instance, the 55 speedup factor was achieved at the expense of 23-fold increase in resource usage[3].

The results of **Table (3)** and **Table (4)** for the case of the Virtex-2Pro platform are plotted in **Figure (3)** and **Figure (4)**, respectively.

[3] See Table (4) for more details.

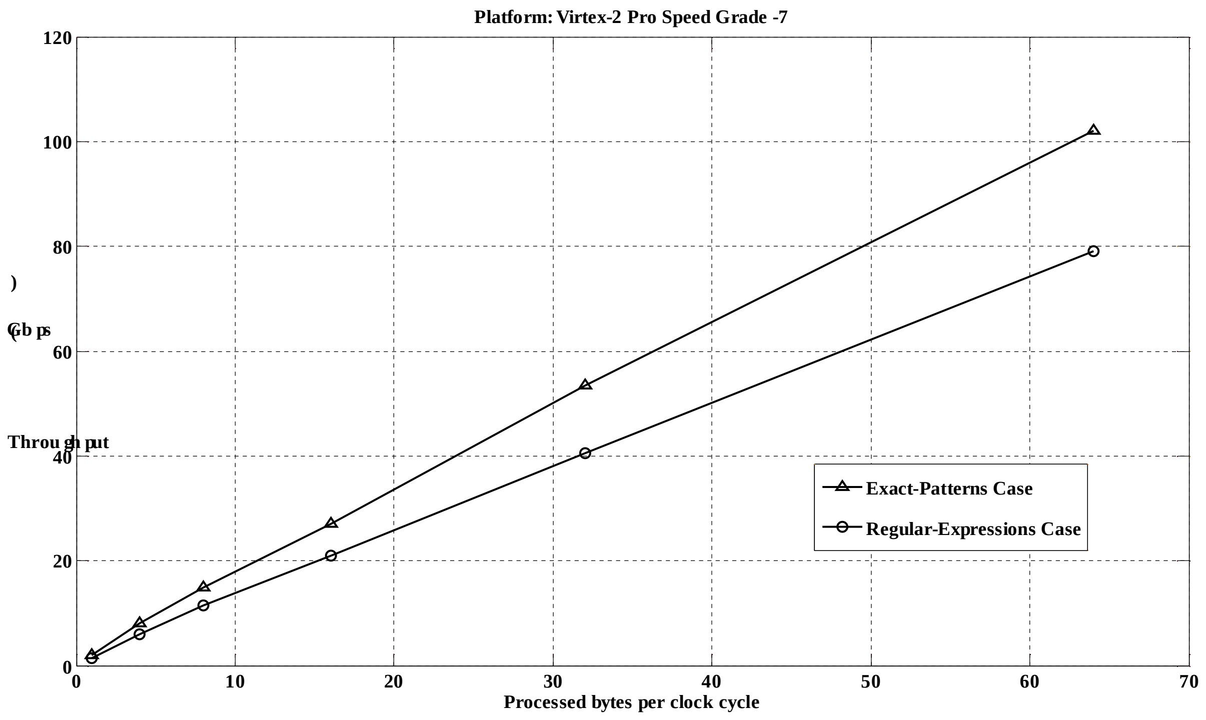

Figure (3): Effect of byte-rate increase on Throughput

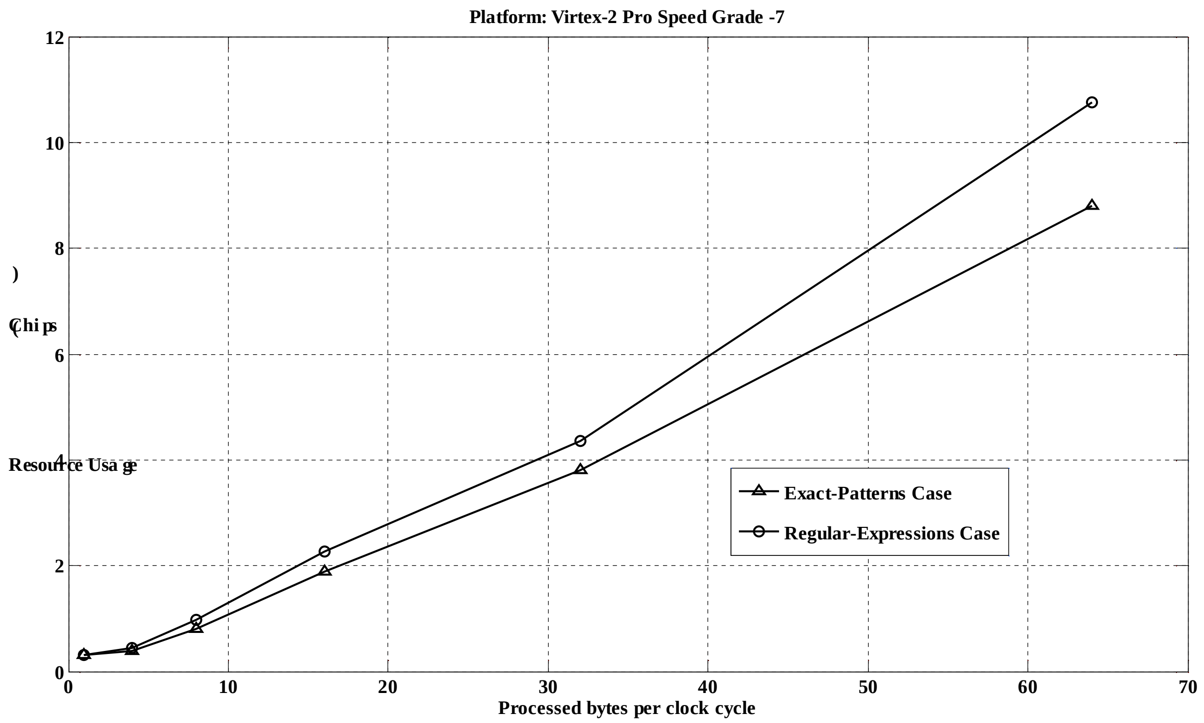

Figure (4): Effect of byte-rate increase on Resource Usage

5. Conclusions and Future Directions:

The focus of this research, so far, has been to design a scalable architecture capable of coping with the increasing demand for high-throughput pattern matching in network IDSes, To this end, we believe that the multi-byte architecture presented here is well posed to meet current and future needs. It is also our view that the additional cost reflected in increased resource usage is acceptable, and justifiable, to attain the required performance levels.

We are currently working on our automation tool to complete the path from regular expression to FPGA configuration bit-stream and bypassing the vendor-supplied, general-purpose synthesis tool. This will serve two purposes. First, it will allow us to optimize the resulting layout to better suit the regular structure of the architecture. And second, it will allow rapid run-time reconfigurable (RTR) of the system when a new rule is added to the rule-set.

Another direction of future research is to conduct a thorough analysis of the overhead inflicted due to the use of a general-purpose FPGA platform in the form of non-utilized resources. It is expected that several optimizations can be made by tailoring the underlying reconfigurable architecture to suit the application. In other words, we will be considering the design of Application-Specific Reconfigurable Architectures (ASRAs).

References:

[1] R. Sidhu and V. K. Prasanna, *Fast Regular Expression Matching Using FPGAs*, IEEE Symposium on Field-Programmable Custom Computing Machines, P. 227-238, 2001

[2] B. L. Hutchings, R. Franklin and D. Carver, *Assisting Network Intrusion Detection with Reconfigurable Hardware*, IEEE Symposium on Field-Programmable Custom Computing Machines, P. 111-120, 2002

[3] Y. H. Cho, S. Navab, W.H. Mangione-Smith, *Specialized Hardware for Deep Network Packet Filtering*, International Conference on Field-Programmable Logic and Applications, P. 452-461, 2002

[4] Christopher R. Clark and David E. Schimmel, *Scalable Pattern Matching for High Speed Networks*, IEEE Symposium on Field-Programmable Custom Computing Machines, P. 249-257, 2004.

[5] Tamer F. Badran, Hany H. Ahmad and Mohamad Abdelgawad, *Network Intrusion Detection: A Review*, Assiut University Conference on Engineering between theory and Practice Conference, P. 172-178, 2007

[6] Ioannis Sourdis, *Efficient and High-Speed FPGA-based String Matching for Packet Inspection*, M.Sc. Thesis, Technical University Of Crete, Electronic And Computer Engineering Department, 2004

Nomenclature:

L_J …	The number of bytes processed per clock cycle
L_P …	The length of the target pattern