Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/120214

Efficient String Matching FPGA for speed up Network Intrusion Detection

J. Armstrong Joseph[1,*], *Reeba Korah*[2] *and S. Salivahanan*[3]

[1] Dept. of ICE, Anna University, Chennai, India.
[2] Alliance College of Engineering and Design, Alliance University, Bangaluru, India.
[3] SSN College of Engineering, Chennai, India.

Received: 7 Jan. 2018, Revised: 19 Feb. 2018, Accepted: 23 Feb. 2018
Published online: 1 Mar. 2018

Abstract: Malicious attacks and threats over network can be identified and prevented by Intrusion detection system (IDS). Essential ability of every intrusion detection system is to search and find packet content that can matches distinguished attacks. An open source Network Intrusion Detection System (NIDS) is Snort that utilizes signatures/rules for detecting irregular network activities. Software-based IDS may not be continued to process all traffic in real-time when network traffic increased. On the other hand, hardware based IDS are best suited for computing and serious processing on network traffic and can keep up high network throughput. This paper, contributes Buffered Boyer-Moore string-matching algorithm using FPGA that drastically increase throughput and improve its performance on hardware implementations. The projected performance as Performance Efficiency Metric (PEM) 21.3 enables system to do 19.2 Gbps of throughput and implies a significant difference obtained when processing large number of payload.

Keywords: Boyer-moore (BM) algorithm, Network intrusion detection, string matching, FPGA

1 Introduction

General attribute of network intrusion detection is to look at packets entering from a network and to scrutinize content of the packet headers and payload. Intrusion detection systems which are more obscured than simple packet classification systems in which one needs to look at the packet content that is payload as well as the packet headers. Pattern matching systems have facility to search for a string or regular expression wherever within the packet. Once a possible threat is found as defined by the matched rule, actions could be taken as either allows or drops the packet. This is used to generate a report detailing the alleged threat. Many algorithms have been applied for packet classification and Pattern matching systems. Well known software based intrusion detection system is being Snort [1, 2]. However, the predictable software solution can have difficulty in keeping up higher network speeds, performance and use. The drawback to the software based IDS is the serial processing the rules. An alternate approach ought to be to use rules being checked in parallel. Such parallel processing may be realized in reconfigurable hardware such as the FPGA (Field Programmable Gate Array). The aim of this proposed work is to study the efficiency of IDS using FPGA that designed by string-matching system with a pre-decoder Finite State Machine (FSM) for use in high-speed network intrusion detection system and to show that this could be applied using current up to date hardware.

2 Related work

Many researchers designed signature (rule) detection architectures based on NFAs and DFAs [3–7] though, they may not give high performance and also process more than one character per cycle easily. Using CAMs and discrete comparators for searching payload against the patterns contained in NIDS rule set [8]. In these studies give the area per slice cost is higher, since they are easy to implement pipeline logic and process multiple bytes per cycle their performance is increased. A technique is applied to increase sharing and reduced designs cost is the use of pre-decoding, it is applied in CAM and regular expression approaches. Sourdis et al. [9] introduced Pre-decoded CAMs for Efficient and

* Corresponding author e-mail: armstrongauphd@gmail.com

High-Speed NIDS Pattern Matching. Dharampurikar et al. [10] presented Bloom Filters with field programmable gate array (FPGA) technology to do string matching which is efficient and very low-cost approach. Michael Attig and Gordon Brebner et al. [11] presented high throughput by increasing data path width. Bande et al. [12] used unique subsequence matching. Hao Wang et al. [13] illustrated MIN-MAX to support matching of regular expressions.

During runtime reconfigurable within programmable device hardware is achieved through either partial reconfiguration or context switching. Course grained reconfigurability [14] is achieved by multiplexing various functional units. Fine-grained reconfigurability requires large routing area so it gives low efficient. Building blocks of FPGA contain more than one LUT, flip-flop and mixture of arithmetic, combinatorial, and multiplexing logic [15]. Dynamically configurable gate array (DCGA) concept [16] is implemented for context switching at run-time reconfigurable system and also included context swapping within an FPGA. This is based on providing a number of resources like ALUs, RAMs, Multipliers, and registers. Partially reconfigurable track the difference based flow on the differences between the designs. By making small changes to a design and then generating the bit streams for the difference. One of dynamically reconfigurable architecture is PipeRench [17] which allows configuration of processing elements (reconfigurable) to change in each execution cycle. This provides a global bus for data transfer. Kilocore KC256 is a commercial version of PipeRench which is a specialized structure for the pipelined execution.

3 Methodology

FPGA-based network intrusion detection system will be rules-based, using rules generated for the software based Snort IDS. Network packets are getting from the network link RJ45 and preprocessed as various rules sets. The preprocessing works is designed for each rule associated with Intrusion detection engine that is secondary FSM. All rules definitions are controlled by a FSM is called as MasterFSM. Each secondary FSM gets control from MasterFSM and classify the packet data into packet header or content of packet (payload). Decoder approach is followed in FSMs. In packet header contains the information about such as protocols, port and source and destination addresses. In payload of packet data contains the rule dentition which is used for searching content stored in DRAM. If it is found match, result out will be delivered or pass the control to next secondary FSM by MasterFSM. Each result is recorded in registers that is called as scoreboard.

Each Intrusion Detection engines are designed by rule definition specified in the rule which is controlled by secondary FSM. They would be set to find payload contents while passing packet data. This can be implemented by comparator and adder. Comparator compares data from DRAM with packet data and delivers result as drop or match. The adder is used for computation of new keys into DRAM. In this paper, Detection engine's string matching implemented by using buffered BM algorithm. Result Output is in the form of alerts, logs and databases. Performance is analyzed in terms of throughput to get efficiently here. As a result Performance Efficiency Metric (PEM) is calculated as follows.

In order to decide how efficient [22] a design is, we calculate the efficiency in terms of performance as

$$\frac{\text{Throughput in Mbit/s}}{\text{Area in slices}}$$

Figure of merit is calculated as, where throughput is defined as

$$\frac{\text{working frequency} \times \text{Number of Bits}}{\text{number of cycles}}$$

Performance Efficiency Metric (PEM) [9] is used to index efficient as defined as

$$\frac{\text{Throughput}}{(\text{Logic Cells} + (\text{MEM bytes}/12))/\text{Non-Meta Character})}$$

where MEM bytes become 0. Number of logic cells calculated as 1.6 $\times$ number of LUTs used in design. We use a metric speedup, is defined by a method for increasing the performance between two systems processing the same instance. But, speedup can be used to explain the effect on performance after any resource development. Using speedup formula as

$$\frac{\text{Latency of old architecture}}{\text{Latency of New architecture}}$$

where Latency of architecture L is the inverse of the execute speed of system. Significant speedup is obtained in overall execution time only for large pattern size. By editing binary configuration file used to increase the speedup in Multicontext FPGA for reconfiguration according to input pattern. Fig. 1 describes block diagram of proposed Intrusion Detection System.

Following subsections given in detailed explanation.

3.1 Preprocessing works for design

From snort database the rule definitions are derived and assigned each rule to design intrusion detection engine. Rule definitions have contents options that are converted from infix notations to postfix notations if necessary. The data values are stored into DRAM by external circuit by vc707 evaluation board.

All snort rule definitions are converted in the form of Hex value content into binary for designing string matching circuit with the help of FSM.

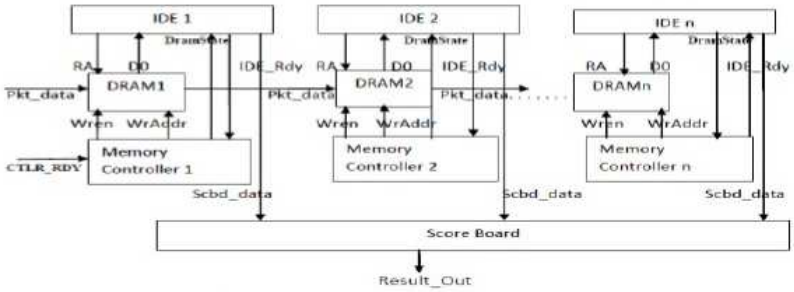


Fig. 1: Block diagram of Proposed Intrusion Detection System.

3.2 Snort

A system based on snort rules optimized for more than thousand need string matching against data of incoming packet.

An example for Snort rule is:
alert tcp any any → 172.198.2.1/32 333 (content: "ieca| 4a4b|"; msg: "mounted success";)

This rule seeks a TCP packet with any source IP is 172.198.2.1 and port no = 333 respectively. To match this rule describes that packet payload contains pattern "ieca | 4a4b| ", in which i, e, c, a are ASCII characters and "4a" and "4b" are quoted within | as Hexadecimal format. Intrusion detection systems are capable to do packet classification and inspection. Their major bottleneck is signature (rule) detection which limits performance of NIDS.

3.3 Intrusion Detection System (IDS)

The IDS simulates snort database contents. Subsets of the snort rule set from snort database is designed for each Intrusion Detection Engine (IDE). Each IDE consists of a primary FSM and secondary FSMs. Primary FSM is designed according to packet header and secondary FSMs is designed for payload that can be inspected to check with packet data. A new packet arrives, the Primary FSM reads in the first rule from the DRAM (Distributed RAM) which has pre-coded values. Once packet is scanned, Primary FSM delivers the result output as matched or drops the packet.

3.4 Intrusion Detection Engine

Each Intrusion Detection Engine is constructed according to snort rule and controlled by FSM. String matching is computed done by logical elements configured as comparator/adder. Comparator compares the specific rule content stored in memory (DRAM) with incoming packet data. Reconfigurable logic elements will act as either comparator or adder based on input pattern. Adder is used to compute new keys in DRAM. After completion of scanning a particular rule on a packet, an IDE revises its equivalent entry in the Scoreboard Register. The Scoreboard maintains a record of the result for each rule.

3.5 Boyer–Moore (BM) algorithm

BM algorithm is an efficient string-matching technique that uses concurrent multi-keyword comparisons. Before the comparisons are performed, it builds tire in reverse according to snort rule set. Tire is tree data structure with a kind of ordered search tree that is used to store a dynamic set or associative array. So, concurrent comparison only requires current packet pointer into trie node. On success, continue down trie and if at leaf, check whether truncated characters match. Otherwise, shift by precomputed shift size on failure pointer. Precomputed shift sizes are computed with two tables. One table is bad-symbol table that indicates how much level to shift based on text's character cause a mismatch. Another is good-suffix table that indicates that how much level to shift based on matched part (suffix) of the pattern. Buffered input is registered by registers.

3.6 Buffered input

In common, BM algorithm is hard to use in pipelined architecture of matching elements. Because of freeze and delay this algorithm, matching processing of new characters are in irregular ways. A partially mismatched pattern occurs due to input text is less than pattern size. This can cause stalling that is unacceptable in array of secondary FSM. This can be rectified by preprocessing that takes alignments allowed with input text over pattern instances are shifted by 16 bits for the 64 bit-per-cycle architecture. A bandwidth of 64 bits per cycle can be achieved with four hardwired 16 bit comparators, allows the running time to be reduced by 4 for an equivalent increased in comparator. A proof of buffer size required to create a string-matching secondary FSM that will not stall. This allows a matching unit to accept one character

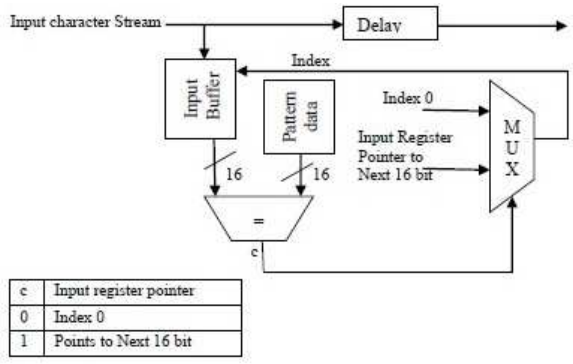


Fig. 2: Architecture of comparator design for buffered input.

each cycle into a buffer size equals to $\log_{\alpha}^{k}$ where $\alpha = (1+\sqrt{5})/2$ and k is pattern size. Solution is to buffer the entire input sequence by registered inputs. But it is cumbersome as input sizes become large. Buffered inputs guaranteed that always one character per cycle could be accepted for searching process simultaneously. Fig. 2 illustrates architecture of comparator design for buffered input. Input is buffered with hardware based BM algorithm to obtain high throughput efficiently.

This proposed paper has presented string matching architecture using buffered input with BM algorithm. The architecture competes with the latest while providing reconfiguration and more efficient hardware use. Table 1 describes Buffered input Boyer-Moore algorithm pseudo-code is as given in Table 1.

3.7 Reconfigurable computing

Accessible Reconfigurable computing (RC) models [20] are formed as either one dimensional or two dimensional arrays of configurable logics cells. They are linked by configurable switches or programming links. The architecture would be graded by its processing ability of each logic cell at various granularity levels such as coarse grained, fine grained. Configuration data could be stored on SRAM/DRAM of FPGA device. Granularity is graded as one-bit output of each Look up Table (LUT). Due to a number of switches used for the programmability creates large delay and large interconnects area. It can be reduced by providing greater data width per LE Array. Single bit output of each Look up Table (LUT) in the LE results in a large interconnects area and large delay due to a number of switches for the programmability. In order to avoid large interconnects area and large delay, provides increased data width per LE. We propose to use an LE with multi bit output LUTs [21]. Since multi bit output LUT has fine grained configurable cell performed with simple logic functions with more complex functions (In this case is DRAM function generator used for computation output). Each logic element (LE) can comprehend two different operations equivalent to six bit

Table 1: Pseudo-code of Buffered input Boyer-Moore algorithm.

```
int tablesize = 256;
int[] table;
String pattern;
Function table(String pattern)
{
table = new int[tablesize];
for (int el = 0; el < tablesize; el ++)
table[el] =-1;
for (int m=0; m < pattern.length(); m++)
table[pattern.char(m)] = m;
/* store pattern character into pattern registers */
}
public int Textsearch(String text, String pattern) {
int pl = pattern.length();
int tl = text.length();
int skip;
for (int l = 0; l < = tl-pl; l += skip) {
skip = 0;
for (int m = pl-1; m > = 0; m–) {
if (pattern.char(m) ! = text.char(l + m)) {
skip = max(1, m-table[text.char(l + m)]);
break;
}
if (skip == 0)
return l;
}
endif (pattern.char(m) == text.char(l + m)){
++ skip;++m;
break;
}
}
if(skip ==m-1)
print ”Match found”;
endif
return tl;
}
```

input to single bit output or five bit input to single bit output. Configuration data could be stored on SRAM/DRAM of FPGA device. Up to thirty two different configurations can store in each memory cell. Like a context switch, LE is reconfigured form comparator to adder in a single cycle and vice versa. The FSM designed to control them as necessary at run-time. In this method parallel processing is achieved by IDS is divided into a series of stages with each stage has separate IDE and DRAM sequentially.

3.8 Scoreboard

Record of the result is maintained for each rule associated with IDE by a unit is called as scoreboard. To maintain IDE search quicker, scoreboard uses modified LFU algorithm is to assign a counter to every memory controller of each IDE block that is loaded into Distributed RAM (DRAM). Each time a reference is made to that trap the counter is increased by one. When

Table 2: PEM Comparison of various FPGA-based pattern matching approaches.

Approaches	Input bits per cycle	Device	Throughput (Gbps)	Logic cells	Eq. Logic cell/char	# chars	PEM
Sourdis et al. PreD-CAM	32	Virtex2–6000	9.708	64,268	3.56	18,036	2.73
Cho et al. Dis. Comp BruteForce	32	Altra EP20K	2.88	17,000	10.6	1,611	0.27
Baker et al. PreD-CAM	8	Virtex2P-100	1.896	6,340	0.32	19,584	5.86
Clark et al. Decoder NFA	32	Virtex2–6000	6.077	54,742	3.1	17,537	1.946
Gokhale et.al. Dis. Comp	32	VirtexE-1000	2.716	9,722	15.2	640	0.14
Sidhu et.al. NFA	8	Virtex-100	0.46	1,920	66	29	0.01
Franklin et al. Comparator NFA	8	VirtexE-2000	0.39	40,232	2.52	16,028	0.16
Moscola et al. DFA	32	VirtexE-2000	1.184	8.134	19.4	420	0.06
Bande et al. NFA	32	Virtex-5 FX100T	5.69	12,733	1.62	5,024	2.24
Hao Wang et al. NFA	8	Virtex-5 LX1107	2.57	8,640	0.33	25,829	7.8
Proposed Method Decoder DFA BM	64	Virtex-7 X485T	19.2	42,182	0.9	46,500	21.3

Table 3: Speedup over other approaches for 8 patterns with input text size varies from 10^3 to 10^5.

Approaches	$T_M + T_{RE} + T_E$ (s)			speedup		
Input Text Size	$n = 10^3$	$n = 10^4$	$n = 10^5$	$n = 10^3$	$n = 10^4$	$n = 10^5$
Proposed Method Decoder DFA BM	0.0018	0.018	0.18	1	1	1
CAD tool [24]	76	76	76.2	$\approx 4X10^6$	$\approx 4X10^6$	$\approx 4X10^6$
Mapping data [25]	0.0218	0.0393	0.2041	12.1	2.1	1.1
Software based snort IDS with BM algorithm	0.03	0.08	0.289	16.6	4.4	3.7

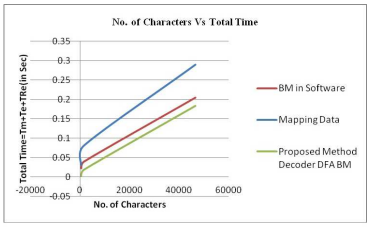
the DRAM reaches capacity and has a new IDE block waiting to be inserted the system will search for memory controller of IDE block with the lowest counter and remove it from the DRAM. A simple modification to LFU that avoids the problem of throwing out the least used page. A redundant *R* bit is assigned to each memory controller while used. The redundant R bit of the oldest memory controller is inspected. If it is 0, the memory controller is both old and unused, so it is replaced immediately from DRAM. If the R bit is 1, the bit is cleared, the memory controller is put onto the end of the list of memory controllers, and its load time is updated as reset to the current time though it had just arrived in DRAM. Then the search continues.

3.9 FPGA Implementation

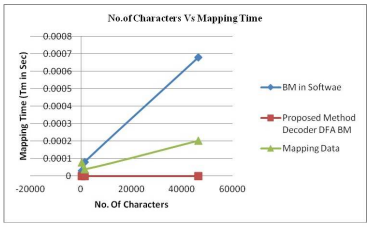
We used tool Vivado v2015.4 (64-bit) for synthesis and place and route this proposed method. The examples in the benchmark suite implemented for Virtex-7 X485T FPGA device Xc7vx485tffg176–1-2. The FPGA has 75,900 'slices', each containing four six-input lookup tables (LUTs) and eight flip-flops(FFs). The target device is the Virtex-7 X485T FPGA device Xc7vx485tffg176–1-2 speed grade [16]. Table 2 Shows that PEM Comparison of different FPGA-based pattern matching approaches. A Non-deterministic Finite Automation (NFA) is mapped on an FPGA [4] achieved low throughput. To build and configure, NFA is complicated. In contrast, discrete comparators [6, 18] are used to obtain higher throughput at the increased area cost. Area cost is reduced by using CAM based solutions to get similar throughput. With the drawbacks of increased area cost and power consumption, proposed pre-Decoder DFA is used to find probable matches on the string set and reduce the total number of comparisons. Our design operates at maximum speed of is 300 MHz while using approx 46500 characters. Design takes 42,182 logic cells and 26,364 LUT and 7,962 Flip-flops. Design execution time is 0.018 second. Our design achieves the best throughput/area efficiency as Performance efficiency Metric (PEM) is 21.3 on FPG and throughput is 19.2 Gbps.

Performance [22] calculated in terms of throughput (Gbps)/area(slices). Two logical cell forms one slice. Higher throughputs achieved by number of bits per cycle processed. By using 64 input bits per cycle achieved throughput is 19.2 Gbps from the proposed method. The performance degradation occurred as either more character increases or as number of states increases.

The devices have on chip DRAM to store a number of configuration context vary from 8 to 256 and switching contexts takes 5 to 100ns. The context switching [23] is like a switching of process on a uniprocessor. Active context (bits stored in all the flip-flops) have saved before switching to a different context. At the same time only one context is active and other contexts are remained. Switched context will execute from where it has stopped earlier. Multicontext FPGA rapidly switches between stored configurations. Our proposed method Decoder DFA BM uses Multicontext FPGA for Logical element reconfiguration. We obtained the performance Software based snort IDS with BM algorithm for searching pattern which is implemented on Pentium 2.8GHz workstation.



(a) No. of characters vs. Total Time.



(b) No. of characters vs. Mapping Time.

Fig. 3

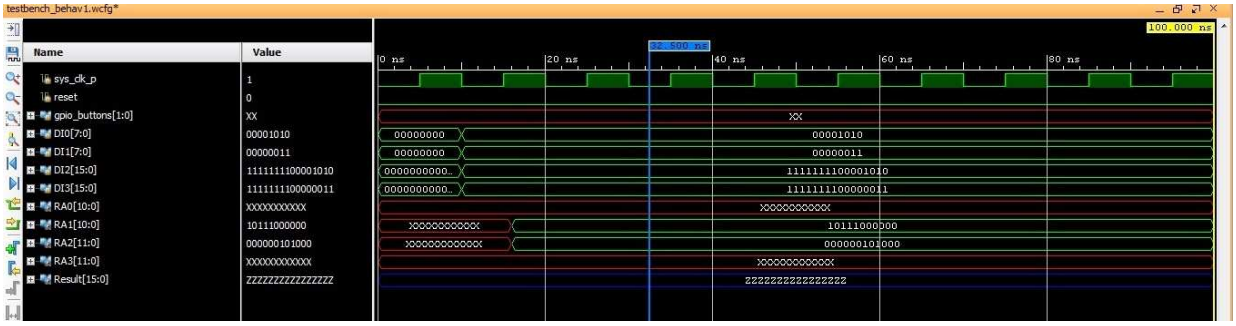


Fig. 4: Simulation Screen shot of the proposed IDS system.

Mapping time $(T_M + T_{RE})$ is taken for Proposed Method Decoder DFA BM approach with other approaches. Where T_M denotes mapping time and T_{RE} denotes reconfiguration time or time required to configuration onto device. T_E is total execution time. Derive the equations T_M, T_{RE} and T_E in terms of in terms of T_{CM} (configuration memory read or write time), T_{clk} (one clock cycle Time) and $S_{\text{OR-GATE}}$ (switching OR gates). The FSM (Finite State Machines) is constructed on a separate context. While executing current context cannot be modify it. Data sharing between contexts are possible on multicontext FPGAs. Reconfiguration is performed via DRAM and writes construction of back edges. Runtime generated FSM is swapped by switching between two contexts. Time calculations are based on the equations:

$$T_{\text{RE}} = (m-1)S_{\text{OR-GATE}}T_{\text{CM}} \quad (1)$$

$$T_{\text{M}} = (4m-1)T_{\text{CM}} + (m+1)T_{\text{CM}} + (7m-4)T_{\text{clk}} \quad (2)$$

$$T_{\text{E}} = (2n-(n/m))T_{\text{clk}} \quad (3)$$

where m is size of pattern and n is size of Text respectively.

Table 3 shows speedup in mapping time (8 patterns context switch) with other approaches. To build logic, template logics are used in CAD tools for each problem once.

Comparison made for implementation of FPGA based string matching approaches. Observed that in [25], Speedups will be getting in very large text searches due to high $T_{\text{M}} + T_{\text{RE}}$ From the result of Table 3, the proposed method shows that speedup significant is obtained about 12.1 to 1.1 while using input text size varies from 10^3 to 10^5 respectively. Fig. 3(a) and Fig. 3(b) shows that comparison of mapping time and total time versus no of characters used in various approaches as in graph respectively. With the help of Eqs. (1) to (3) we obtained a speedup significant when processing no of characters varies from 10^3 to 10^5 respectively.

Proposed method is designed by a device Virtex-7 X485T and it has configured by dynamic reconfiguration using Xilinx Vivado v2014.2. The result shows that significant speedup is obtained while using input text size varies from 10^3 to 10^5 by getting different mapping time and reasonable reconfiguration time over various existing approaches.

Figs. 4 and 5 show screenshots of simulations output and device implementation of our system respectively. It clearly indicates that successive matching signature output and its corresponding resources utilized in this design.

4 Conclusion and Future work

In this paper the features of Xilinx FPGA are used to map software based Snort IDS into hardware based snort IDS using FPGA and the amount of logic resources decreased. The paper shows that proposed system can maintain throughput of 19.2 Gbps with performance in terms of Performance Efficiency Metric (PEM) 21.3. Moreover, the system is implemented on device Virtex-7 X485T supporting board for networking application. The proposed method based on pre decoded DFA of the

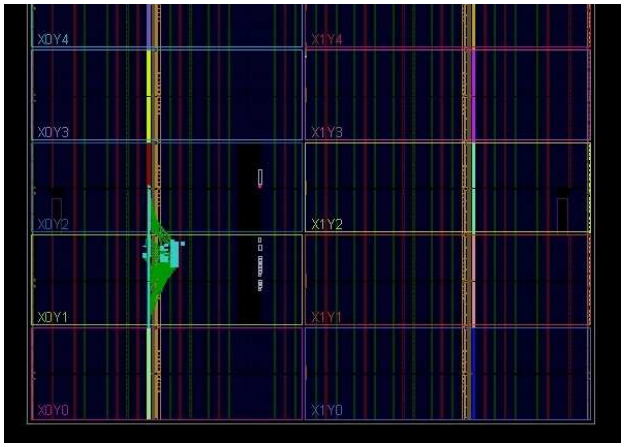


Fig. 5: Device Implementation Screen shot of the proposed IDS system after place and route.

incoming packets from Snort ruleset. In this application, FSM constructed to use control memory and string matching. Results show that significant speedup is obtained in mapping time and total time over various existing approaches. Upcoming work is to preprocess the snort rule set by an optimizer [19] that can determine their interdependencies of rules. This will reduce number of rues, so searching pattern is very easy.

References

[1] Sourcefire, Snort: The Open Source Network Intrusion Detection System, http://www.snort.org, 2017.
[2] Haoyu Song, T. Sproull, M. Attig and J. Lockwood, Snort off loader: a reconfigurable hardware NIDS filter, Field Programmable Logic and Applications, 2005. International Conference on, 24–26 Aug. 2005.
[3] Z.K. Baker and V.K. Prasanna, Automatic Synthesis of Efficient Intrusion Detection Systems on FPGAs. In: Proceedings of 14th International Conference on Field Programmable Logic and Applications, August 2004.
[4] 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.
[5] J. Moscola, J. Lockwood, R.P. Loui, and M. Pachos, Implementation of a Content-Scanning Module for an Internet Firewall. In: IEEE Symposium on Field-Programmable Custom Computing Machines, April 2003.
[6] R. Sidhu and V.K. Prasanna, Fast Regular Expression Matching using FPGAs. In: IEEE Symposium on Field-Programmable Custom Computing Machines, April 2001.
[7] C.R. Clark and D.E. Schimmel, Scalable Parallel Pattern-Matching on High- Speed Networks. In: IEEE Symposium on Field-Programmable Custom Computing Machines, April 2004.
[8] 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 Int. Conference on Field Programmable Logic and Applications, 2002.
[9] I. Sourdis and D. Pnevmatikatos. Pre-decoded CAMs for Efficient and High- Speed NIDS Pattern Matching. In: IEEE Symposium on Field-Programmable Custom Computing Machines, April 2004.
[10] M. Attig, S. Dharmapurikar, and J. Lockwood. Implementation Results of Bloom Filters for String Matching, In: IEEE Symposium on Field-Programmable Custom Computing Machines, April 2004.
[11] Michael Attig and Gordon Brebner, 400 Gb/s Programmable Packet Parsing on a Single FPGA, Xilinx Labs 2100 Logic Drive.
[12] J.M. Bande Serrano, J.H. Palancar, String alignment pre-detection using unique subsequences for FPGA-based network intrusion detection, Computer Communications, **35** (2012) 720–728.
[13] Hao Wang, Gabe Knezek, and Jyh-Charn Liu, MIN-MAX: A Counter-Based Algorithm for Regular Expression Matching, IEEE Transactions on Parallel and Distributed Systems, **24(1)**, January 2013.
[14] T. Kitaoka, H. Amano, and K. Anjo, Reducing the configuration loading time of a coarse grain multicontext reconfigurable device, Proc. FPL (LNCS2778), pp.171–180, 2003.
[15] Anupam Chattopadhyay, Ingredients of Adaptability: A Survey of Reconfigurable Processors, Hindawi Publishing Corporation, VLSI Design, Volume 2013, Article ID 683615.
[16] The Xilinx Corporation, Virtex 7 Series FPGA Devices, 2016, http://www.xilinx.com
[17] S.C. Goldstein et al., PipeRench: A Coprocessor for Streaming Multimedia Acceleration; Proc. ISCA'99, Atlanta, May 2–4, 1999.
[18] Hao Chen, Yu Chen and Douglas H. Summerville, A Survey on the Application of FPGAs for Network Infrastructure Security, In: IEEE Communications Surveys & Tutorials, **13(4)**, 2011.
[19] T.T. Hieu. T.N. Thinh, T.H. Vu. and S. Tomiyama, Optimization of regular expression processing circuits for NIDS on FPGA. In: Proceeding of the Second International Networking and Computing Conference. 2011, pp. 105–112.
[20] Wai Kai Chen, The Electrical Engineering Handbook, Elsevier Academic Press. ISBN: 0–12-170960–4.
[21] Huesung Kim, A.K. Somani and A. Tyagi, A reconfigurable multifunction computing cache architecture, Aug 2001, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, **9(4)**, 509–523.
[22] H. Mahdizadeh, and M. Masoumi, Novel Architecture for Efficient FPGA Implementation of Elliptic Curve Cryptographic Processor Over GF (2^{163}), IEEE Trans. VLSI Systems, **21(12)**, 2330–2333, 2013.
[23] A. Beasley, L. Walker and C. Clarke, Developing and Implementing Dynamic Partial Reconfiguration for Pre-Emptible Context Switching and Continuous End-To-End Dataflow Applications, In: Altera SoC Developers Forum, 2015–14.
[24] P. Lysaght, B. Blodget, J. Mason, J. Young, and Bridgeford, Invited Paper: Enhanced Architectures, Design Methodologies and CAD Tools for Dynamic Reconfiguration of Xilinx FPGAS" International Conference on Field Programmable Logic and Applications, 2006.

[25] Krishna N. Vikram, and Vinita Vasudevan, Mapping Data-Parallel Tasks Onto Partially Reconfigurable Hybrid Processor Architectures, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, **14(9)**, 2006.

J. Armstrong Joseph is research scholar in Anna University in Chennai. He obtained B.E. degree in Electronics and Communication Engineering from National Engineering College, M.E. degree in Computer Science and Engineering from Manonmaniam Sundaranar University, Tirunelveli. His areas of interest in research are VLSI, Computer Networks and Information Security.

Reeba Korah heads the Department of Electronics and Communication Engineering of Alliance University, Bangalore. She has a vast experience of over 24 years in the field of engineering, academics, administration and active research. Prof. Korah is an alumnus of Marathwada University, Maharashtra and Anna University, Chennai. Her technical expertise spans VLSI design, image and video processing and wireless sensor networks.

S. Salivahanan is the Principal of SSN College of Engineering, Chennai, since July 2003. He obtained B.E. degree in Electronics and Communication Engineering from PSG College of Technology, Coimbatore, M.E. degree in Communication Systems from NIT, Trichy and Ph.D. from Madurai Kamaraj University, in the area of Microwave Integrated Circuits. His areas of interest in research are Microwave Integrated Circuits, Low and High Frequency EM Fields and Digital Signal Processing.