ISSN 22773061

Logical Circuits for Extended Content Matching in Hardware Based NIDPS

Dejan Georgiev, Aristotel Tentov
Faculty of Electrical Engineering and Information Technologies - Skopje, Macedonia
dejan@inbox.com
Faculty of Electrical Engineering and Information Technologies - Skopje, Macedonia
toto@feit.ukim.edu.mk

ABSTRACT

In this paper we present logical circuits for efficient detection of rolled out contents. As network speed increases and security matters there is a demand for implementation of hardware based Network Intrusion Detection and Prevention Systems (NIDPS). On the other hand hardware based NIDPS are lacking the flexibility of detection of so named "evasion" techniques. Here we present simple but efficient enhancement to content matching in hardware with minimal basic memory elements (flip-flops) used.

Indexing terms/Keywords

Content matching, character repetition, quantified repetitions, NFA, Kleene plus

Academic Discipline And Sub-Disciplines

Computer technology / Digital logic

SUBJECT CLASSIFICATION

Logical circuits design

TYPE (METHOD/APPROACH)

Quasi-Experimental / Simulation

Council for Innovative Research
Peer Review Research Publishing System
Journal: INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY
Vol 7, No 3
editor@cirworld.com
www.cirworld.com, member.cirworld.com

INTRODUCTION

Content matching is a powerful option used at software based NIDPS like for example the open-source project Snort [3] . Content option inspects every byte of the package payload in aim to accurate detect the known signature or a pattern of a network attack. The process of deep-package inspection is computation intensive and exhaustive one especially in high loaded network traffic. Hardware approach is a feasible solution to overcome the speed bottleneck. One of the drawbacks of hardware implementation is the impossibility to adopt the system to the variations of the previously known attack. The attackers use clever modification of the signature, mostly in the content pattern of the attack to avoid the NIDPS. For example , the simple Snort rule

alert tcp any any -> any any (msg:"Evasion";content: "abcd";)

could be modified by repeating one or more of the characters in the content option

alert tcp any any -> any any (msg:"Evasion";content: "abbcd";)

These methods are named evasion techniques. Content extension could be achieved by insertion of a characters or by repeating of the characters. In this paper we focus on the later. At first glance hardware realization as logical circuit will require adding an extra components for detection the "extended" characters. Formally described as Final State Machine (FSM) the logical circuit accepting the language *L(M)* should be able to detect all the variations of the same language $L(M')$ such as if M is defined over the input alphabet Σ then M' is defined over very same Σ as M. For example, if the content to be detected is "abcd" the machine ought to detect all the rolled out variations of the content including all the characters in same order. In reality, given pattern "abcd" will convert to its extended version "abbcccd" etc. We start with section 2 representing brief review of related work and some basic principles in the scope. Section 3 explains the proposed method for endless extension i.e. repetition of the basic content characters one, more or non-limited number. In section 4 we present hardware circuit for detection of extended version with exact or known number of repetitions. Results and conclusion are given in section 5 and 6.

BACKGROUND

Pattern matching is important problem in many areas including string finding algorithms, biosequence search and DNA matching and of course network intrusion detection and prevention systems (NIDPS). One of the major approaches to an approximate content matching in hardware is by using regular languages and regular expressions. A regular language is a set of finite sequences of symbols from a finite alphabet. For example, over given alphabet Σ={a,b} could be defined the regular language L={a,ab,abb,abbb..} or L={b,ab,aab,aaab...}. A regular expression is a declarative way of expressing the string contained in such a language [4] . Recalling the theory regular language can be recognized by *deterministic finite state automata* (DFA) or non-*deterministic finite state automata* (NFA). It is known that DFA's have no more than one active state in parallel and only one outgoing edge for a given symbol. As such DFA are more appropriate for software implementation. But DFA have one main disadvantage compared to NFA. Namely DFA suffer from state explosions. In hardware. e.g. FPGA NFA's have no conflict with concurrent sates realizations. The first to use nondeterministic state machine on programmable logic are the authors of [1] . They have used the so called technique of One-Hot-Encoding (OHE) scheme. At OHE each state is represented by a flip-flop as the most basic memory element where active state is represented by 1 and inactive by 0. Based on this research an efficient design for complex pattern matching using NFA circuits have been presented by [5] . The same paper has introduced the 8-to256 decoder where instead of broadcasting the 8-bit character this approach sends 1-bit output from the decoder. Extensions to the basic regular expressions containing wildcard ('.', '?', '+', '*') and most similar to ours approach have been presented by [2] .

REPETITIONS DETECTION

The dragged version of a content arise from basic pattern by repeating one, more or all the characters. In this chapter will be observed unspecified and unlimited number of repetition. As it was early mentioned extended content is mostly used by evasion techniques to avoid security policies. The detection requires flexibility and adoption to the raised out pattern. The last is a special difficulty at hardware realizations. Fortunately, symbol repetition could be described with regular languages using regular operator (*) - matches previous item zero or more times,(+) -matches previous item one or more times. On the other hand NFA are appropriate mechanisms to recognize regular languages and regular expressions. Based on the proposed concept for realization of Kleene closure in logical circuit [1] on Figure 1 we present NFA diagram for regular operator "+" or also named Kleene plus.

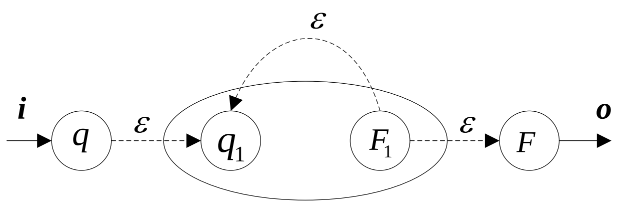

Fig.1 NFA for operator "+"

ISSN 22773061

Obviously ε represent ε-transitions i.e. unconditional transitions of NFA. ***i*** denotes the "token" indicating previously recognized character and with **o** is given the pass to the next character to be detected. Mapped in hardware Fig.1 will be represented by a basic memory element Flip-Flop (FF) and logical ports as shown on Fig. 2.

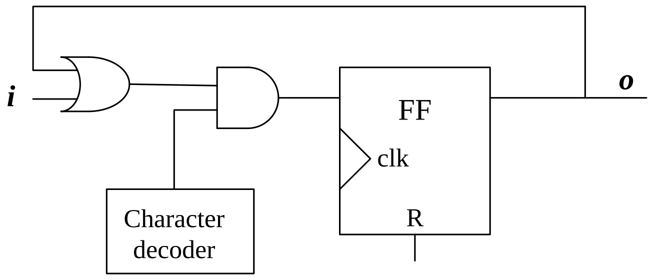

Fig.2 Logical circuit for operator "+"

Effective and area reduced character decoder was presented by [5] known as 8-to-256 decoder. Another solution is the simplest 8-to-1 character decoder i.e. eight input logical AND port as shown on Fig.3

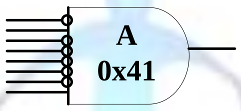

Fig.3 Simple 8-to-1 character decoder (comparator)

Content matching of full sequence and its extended version in hardware could be realized by combining more basic logical circuits in a pipeline presented on Fig.4. For example, for given content "abc" its dragged variant can be overwritten as regular expression "(a+)(b+)(c+)". The system design is capable to detect all the content variants derived from "abc", like for example "aabc","aabbcc", "ab....bc" etc.

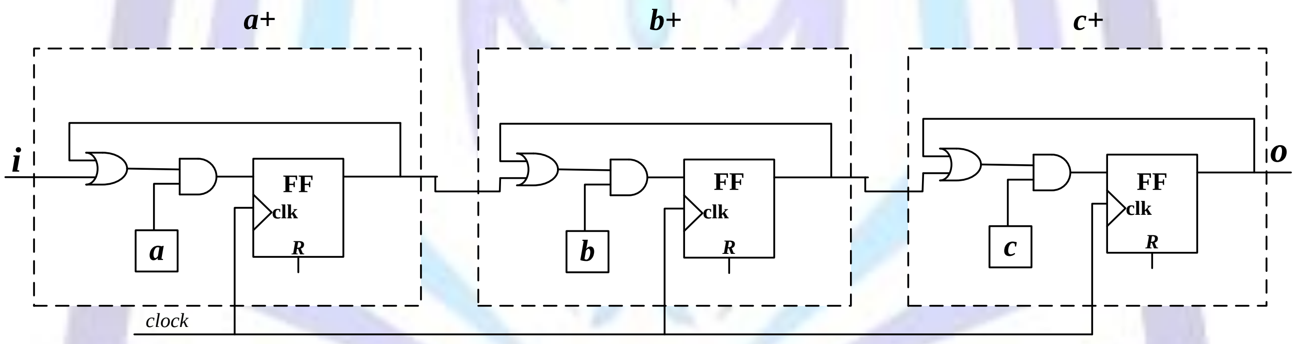

Fig.4 Pipeline for detection of (a+)(b+)(c+)

It worth noting that the logical circuit concept show on Fig.2 could be easily extended to a group of characters detection e.g. (abc)+.

QUANTIFIED REPETITIONS

Practically the proposed design in section 3 is aimed to detect apriority unspecified number of character repetition even infinitely repetitions. On the other hand many of rules from Snort rule set contain exact or constraint number of character repetitions. The most often used operators in that context are Exactly {N},AtLeast{N,} and Between {N,M} number of repetition e.g. b{10}, b{100,} or b{5,7}. NFA circuit for arbitrary number of repetition detection specified by either lower bound or upper bound have been presented by [5] as bounded-length wildcards. It their design the number of used memory elements (FF) is equal to the boundaries defined. It is area expensive design for enormous number of repetitions. Another paper [2] presents Exact {N} matching using bit counter and a sift register. Without area reduction techniques ,like using SLR16 instead in sift-register their approach requires more logic elements compared to [5] .

Our intention is to reduce the number of memory elements implemented retaining the effectiveness of previously proposed designs. Taking in consideration the basic principle of exact and constraint repetitions is counting the number of the repetitions a standard sequential components could be used simulating a Final State Machine (FSM). In our design we will use more compact state encoding than the one-hot-encoding at sift-registers, thus reducing the number of FF used. We use simple binary up counter. Fig.5 represents logical circuit for Exact {N} single character detection.

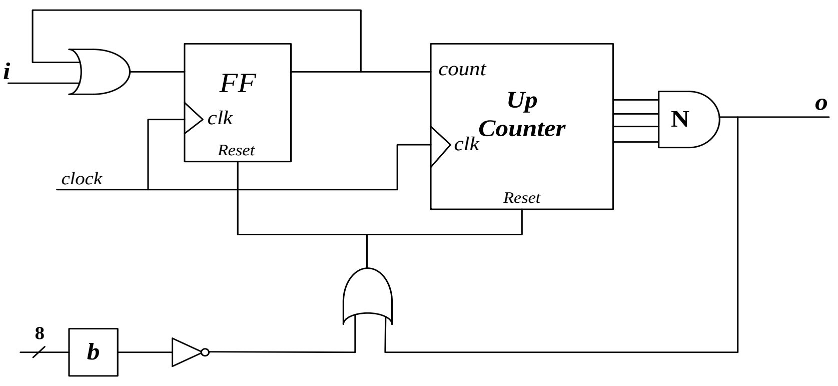

Fig.5 Exact {N} repetitions matching of character b

The input token "i" is received from the previously matched character. FF acts as switch ON/OFF to the up counter. It should be noticed the FF concept design is the NFA presented on Fig.1 i.e. the "+" operator. Once the input token is received FF will remain in ON position until positive Reset signal. While FF in ON position the counter counts up by a binary value starting from 1 instead of 0 since the first count is performed by the FF. The circuit is stepping up for a bit position as long as character decoder detects "b" signal at its inputs. When first mismatch occurs the circuit is reset. At some point the up counter reaches N value generating one clock long output "1" and resets the FF and Up counter preventing the circuit from unnecessary cycling regardless of the input character match. Thus created circuit will require $\log_2 N+1$ basic memory elements. Also the number of inputs to the AND port is given by $\log_2 N$.

Similar circuit design with minor changes can be used for AtLeast{N,} detection. Instead of reset signal conducted to the up counter when N is reached it will only reset the FF thus preventing the counts up and retaining the output signal to high value "1" until a mismatch occur and resets the whole circuit.

The Between{N,M} detection we achieve with slight modification of circuit explained on Fig.5. In that aim we use logical circuit for Kleene closure (*) introduced by [1] .

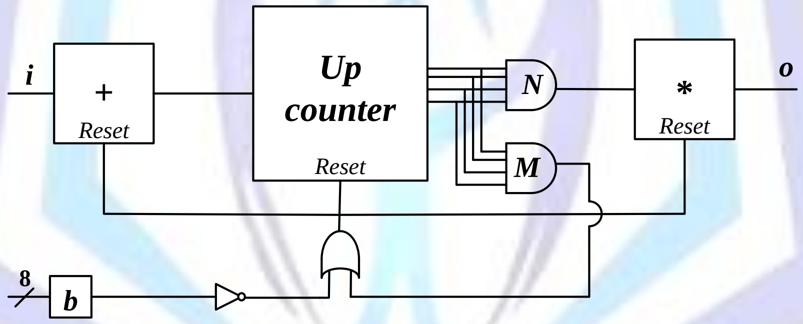

Fig.6 Between {N,M} repetition matching for character b

The positive recognition of previous character is received at FF for "+" operator. It outputs "1" as long as reset signal is set to "0" thus forcing the up counter to increase its value with every clock signal. On the output we add "*" operator performing the same function as "+" except the extra feature "zero or more match". The output ***o*** is active between count N and M. When M is achieved or there is a character mismatch all the elements of the circuit are reset.

So far, quantified recognition refer to a single character. There are many rules that include patterns with blocks of characters e.g. (abc){10}. Combining the principle of unlimited block detection and quantified detection we extend the circuit to match group of characters repeated constraint number in a pattern. Fig.7 displays our approach.

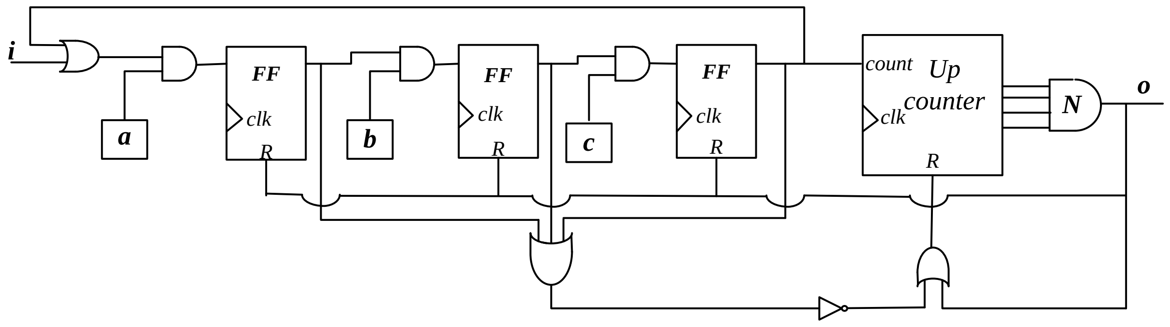

Fig.7 Logical circuit for block detection of "(abc){N}"

The previous detection is transferred as a token from ***i.*** Since entered in the block the token is circulating as long as there is a successive match of the pattern. Every single positive match cause the up counter to increase its value for one bit on clock period equal of the length of the sequence to be match. The counter is triggered on positive *count* signal . Once the token is lost in the block i.e. there is a negative match the counter is reset on the first next rising clock edge. Also the counter output value is set to 0 on achieving the binary value of N. The three input OR gate on Fig.7 confirm the existence of the token in the block. Finally , after N successive repetition of the sequence the token is passed to the output for the next character match.

PERFORMANCES AND RESULTS

Logical operators "+" and "*" require only one FF for their realization. The up counter is implemented with $\log_2 N$ i.e. $\log_2 M$ memory elements (FF) where N<M. Compared to the previous work , without area reduction we decreased the number the number of flip-flops at least by N/$\log_2 N$. In terms of processing speed the proposed implementations are capable to accept one input character at a given time yielding direct correlation to the main speed of the implementation board (FPGA). For example , a throughput of 1Gbps can be acquired on 125 MHz board. Fig. 8 shows simulation results run on Isim (x0.61xd) simulator for an input sequence of "a" character the input ***i*** and the output pulse ***o*** for Exact{N} matching i.e a{3}**.**

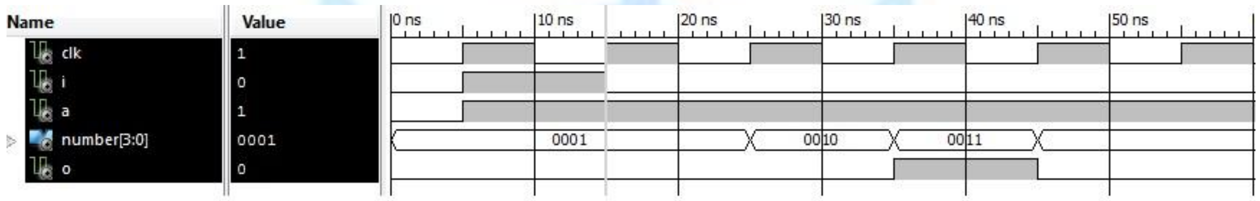

Fig.8 Simulation results for a{3}

Clock period is set to 10 ns. After three matches of "a" character the output is active for one clock period of time.

Similar results are presented on Fig. 9 for a block of characters (ab){3}. It is evident the input sequence "abababab....". After third successive input of the pattern "ab" the output of the logical circuit is set to '1' for a one clock period.

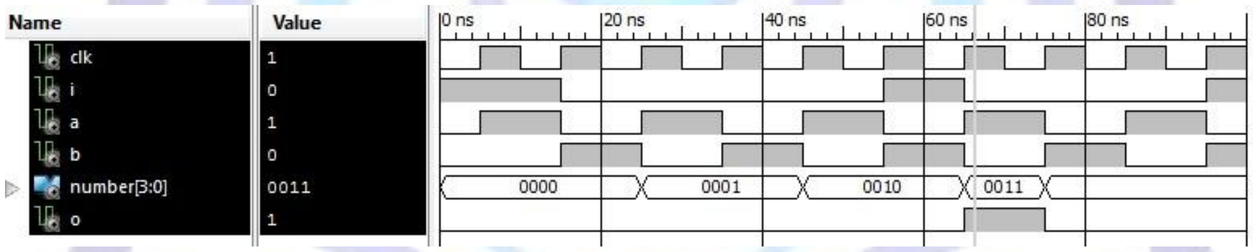

Fig.9 Simulation results for (ab){3}

CONCLUSION

Content matching in hardware is an emerge theme since the 1+ Gbps network speed became standard and security ricks growth. Network intrusion detection and prevention systems in software are superior in flexibility but limited in processing throughput. In this paper we present beyond software feature enhancement to the hardware content detection. Namely evasion techniques are awkward to fit even in software when the number of repetitions is a priory unknown. Character repetition i.e. extended versions of a content is basic principle of the evasion techniques. Here we present simple logical circuits intended to detect one, more, or no repetition with the same component, thus confirming the possibility of hardware realization of any kind of a character(s) repetition. Compared to the previous work here a memory elements reduction was presented and enhancement to a block detection is proposed. A lot of work still can be done, as for example implementation of more complex block expressions. Yet another challenge is parallel processing.

REFERENCES

[1] Reetinder Sidhu, Viktor K. Prasanna, "Fast regular expression matching using FPGA", University of Southern California, Los Angeles CA 90089

[2] Ioannis Sourdis, Stamatis Vassilaidis, Joao Bispo, Joao M.P. Cardoso "Regular expression matching in reconfigurable hardware", Journal of Signal Processing Systems 51,99-121,2008

[3] SNORT (www.snort.org)

[4] Lous Woods "FPGA-Accelerated pattern matching over stream of events", Swiss federal institute of technology - Zurich

[5] Christopher R. Clark and David E. Schimmel , "Efficient reconfigurable logic circuits for matching complex network intrusion detection patterns", In proceeding of International conference on field-programmable logic and applications (FPL), Lisbon,Portugal, September 2003

[6] Enoch O. Hwang ,"Digital Logic and Microprocessor design", La Sierra University, Riverside, 2005

[7] Young H. Cho, William H. Mangione-Smith, "Deep network packet filter design for reconfigurable devices", ACM Transaction on embedded computing systems, vol. 7 ,no.2 article 21, publication date: February 2008

[8] James Moscola, Young H. Cho, John W. Lockwood, "A scalable hybrid regular expression pattern matcher", Department of computer science and engineering, Washington University, St. Louis, Missouri 63130

Author’s

Dejan Georgiev was born in Radovis ,Republic of Macedonia in 1981. He received 5-year engineering degree in elctronics and telecommunications and his Master Degree in telecommunications from Faculty of Electrotechnical Engineering and Information Technologies - Skopje,R.Macedonia. He is currently working towards his PhD in Computer Science and Computer Engineering. He has work experience at home and international telecommunication company and governmental agencies.

Aristotel Tentov,PhD is Full Professor at University "St. Kiril i Metodij" in Skopje, Faculty of Electrical Engineering, Computer Science Department, Skopje, Republic of Macedonia. He has completed his PhD in Computer Science in 1994. Dr Aristotel Tentov published as an author or as a coauthor more than 20 scientific papers on conferences, symposiums and journals. Beside that, he was an author or a coauthor on more than 20 technical reports or projects. Dr. Tentov is engaged in several research areas: Computer Architectures, wired,wireless and mobile networking, Design of Integrated Circuits, Multiprocessor and Multicore Systems, RFID devices and environments etc.