ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 5, No. 1, 2020

PARALLEL COMBINING DIFFERENT APPROACHES TO MULTI-PATTERN MATCHING FOR FPGA-BASED SECURITY SYSTEMS

Sergii Hilgurt

Pukhov Institute for Modelling in Energy Engineering, Ukraine.
Authors' e-mail: *hilgurt@ukr.net*

Submitted on 10.05.2020

***Abstract*: The multi-pattern matching is a fundamental technique found in applications like a network intrusion detection system, anti-virus, anti-worms and other signature-based information security tools. Due to rising traffic rates, increasing number and sophistication of attacks and the collapse of Moore's law, traditional software solutions can no longer keep up. Therefore, hardware approaches are frequently being used by developers to accelerate pattern matching. Reconfigurable FPGA-based devices, providing the flexibility of software and the near-ASIC performance, have become increasingly popular for this purpose. Hence, increasing the efficiency of reconfigurable information security tools is a scientific issue now.**

Many different approaches to constructing hardware matching circuits on FPGAs are known. The most widely used of them are based on discrete comparators, hash-functions and finite automata. Each approach possesses its own pros and cons. None of them still became the leading one.

In this paper, a method to combine several different approaches to enforce their advantages has been developed. An analytical technique to quickly advance estimate the resource costs of each matching scheme without need to compile FPGA project has been proposed. It allows to apply optimization procedures to near-optimally split the set of pattern between different approaches in acceptable time.

***Index Terms*: combining approaches, DPI, FPGA, information security, multi-pattern matching, signature.**

I. INTRODUCTION

The massive propagation of Internet and network applications, coupled with the widespread availability of system hacks and viruses, have made the importance of network security more significant. Increasing number and sophistication of attacks against the network infrastructure necessitates more robust security solutions. Despite the great progress that has been made, there is an ever-widening performance gap between the processing requirements of security tools and their software implementations. The speed limitations of sequential software execution and the increase in network throughput also contribute to the widening of this gap [1].

Field Programmable Gate Array (FPGA) devices have commonly been proposed because they feature both the flexibility of software and the high performance of specialized hardware [2], [3]. So the reconfigurable accelerators became a suitable and popular hardware platform for many security applications, including network intrusion detection systems (NIDS), anti-virus, anti-worms and other signature-based information security tools, which have to scan the incoming data in real time. The computation-intensive multi-pattern string matching task is a major performance bottleneck in such systems, known from the beginning of millennium [4], [5].

In fact, not only signature-based systems are used for network security now. Anomaly detection, fuzzy logic, machine learning, deep learning and many other directions have been actively developing lately [6]. Nevertheless, all of them still suffer from respectively high rate of false positive and, especially, false negative mismatches.

So, in this paper, the state-of-the-art FPGA-based multi-pattern string matching technical solutions have been analyzed, experience of many developers has been studied.

Each of known approaches to build an FPGA-based scheme aimed to fulfill multi-pattern matching has its own strengths and weaknesses. None of them demonstrates key advantages over others, a solution effective enough has not yet been found.

To resolve this contradiction a method to combine several different approaches in one device to enforce their advantages has been developed, the corresponding hardware structure has been constructed. To implement this it was necessary to investigate the features of every approach in terms of resource costs, speed/throughput parameters, functional characteristics and scaling parameters, and also to formalize this knowledge.

II. FPGA-BASED NETWORK INTRUSION DETECTION SYSTEMS

Historically the first and, consequently, the most studied FPGA-based tools of information security were intrusion detection systems (IDS) [7]. Therefore, without losing the generality of reasoning, consider the typical functions and efficiency parameters of reconfigurable security systems on the example of IDS.

Depending on the protected object, there are IDS, which protect individual computers and analyze network traffic packets of the entire local network. As the use of hardware solutions is more efficient for the second type of tools – network intrusion detection systems (NIDS), we consider just such systems.

A. THE STRUCTURE OF RECONFIGURABLE NIDS

The generalized structure and the composition of the FPGA-based network intrusion detection system, composed as a result of analysis of related works, is presented in Fig. 1.

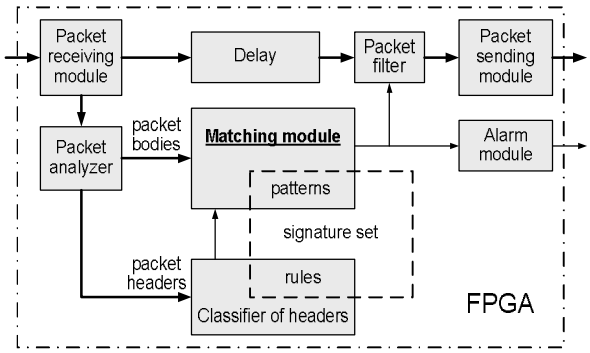


Fig. 1. The generalized structure of the FPGA-based network intrusion detection system

The Matching module (MM) is the most important component of NIDS, which characteristics essentially affect on the performance parameters of the system as a whole [8], This module solves a computationally complex problem of multi-pattern string matching, i.e. checks the content of network packets against certain sequences of symbols, so-called *patterns*, which are parts of the signatures – the descriptions of the known attacks.

Signature databases of modern security protection tools contain from tens of thousands to hundreds of thousands (for NIDS), even millions (for anti-virus) of patterns that need to be detected simultaneously in the input data at wire speed.

B. EFFICIENCY PARAMETERS OF RECONFIGURABLE NIDS

The main parameters (or indicators) of the effectiveness of reconfigurable NIDS which is a typical signature-based security tool can be divided into three groups [9]:

- cost parameters;
- parameters of speed, or performance parameters;
- functional parameters.

Cost indicators are as follows: the amount of logical resources of programmable logic needed to create a digital circuit, memory costs (external to the FPGA chip, internal memory of FPGA – block memory (BRAM) and distributed memory (flip-flops in logical cells)), as well as other costs, which make up the total cost of ownership, including the development, manufacture and programming costs.

Performance parameters include: the volume of the signature database (i.e. the number of pattern to be recognized), the speed of the system (which is defines as either the delay of data propagation from input to output or as bandwidth), and the predictability of speed as well.

An important intermediate metric that links speed and cost characteristics is scalability – the ability to increase performance without excessive resource costs. There are three types of scalability: by the bandwidth, by the pattern set size, and by the pattern length.

Functional indicators include: the ability of NIDS to work in the mode of network intrusion prevention system (NIPS), the ability to dynamically update the patterns without interrupting the matching process, the ability to counter attacks targeted at the NIDS itself, and others.

III. COMPARISON OF THE MAIN APPROACHES TO THE BUILDING OF RECONFIGURING MATCHING MODULES

The analysis of numerical developments of researchers from all over the world shows that when creating NIDS, the best abilities were demonstrated by three approaches which use the following technical solutions based on the corresponding technologies:

- content addressable memory (CAM) based on discrete comparators (DC) [10]–[15];
- Bloom filter (BF) based on hash-functions (HF) [16]–[21];
- Aho-Corasick algorithm (AC) based on finite automata (FA) [22][27].

A detailed study of each approach, which was conducted by the author in [28], [29] and [30] respectively, allows to compare their features and technical capabilities using the parameters discussed above. The Table 1 shows the results of a comparative analysis of these approaches.

As we can see, each of these approaches has its own pros and cons. And none of them fully meet the requirements for the reconfigurable signature-based security systems.

For example, DC and based on them CAM modifications provide maximum performance, but are very expensive in comparison with other approaches in terms of hardware resources and electricity consumption, they also lose in scaling. The Bloom filter is more effective by resources and scalable, but imposes restrictions on the length of the patterns; it also requires additional costs to check the obtained results due to its not eliminable inherent percent of false positive recognition errors. Finite automata are modest in terms of logic consumption, provide stable but relatively low bandwidth, are difficult to build, and lead to an “explosive” increase in memory costs for large signature databases.

The lack of a leading direction that would outperform competitive solutions in all respects makes the developers offer numerous modifications of the main approaches, trying to overcome their shortcomings. But these attempts predominantly are heuristic, unsystematic. A lack of formalization and generalization does not allow to shift the problem from engineering level to the scientific one.

Therefore, based on the study and systematization of existing experience in the construction of reconfigurable NIDS, a method of formalizing the idea of combining different approaches in order to maximize the effectiveness of each was proposed by author. Let us consider this method.

IV. PARALLEL COMBINING METHOD

The essence of Parallel Combining Method (PCM) is to split the pattern set, which should be analyzed by the MM, into subsets when simultaneously synthesizing the same number of matching units (MU). Each of the latter most effectively matches patterns of the corresponding subset, thereby maximizing the benefits of its approach. The highest efficiency is being achieved by covering both the splitting process and the choice of technical solution modification for each MU with a common optimization procedure.

C. THEORETICAL BACKGROUND

The background of the PCM method is in the fact that the patterns included in the signature database differ in length and self-similarity. Hence, the processing efficiencies also differ and depend on the approach of the corresponding scheme.

Another factor in favor of this method is a fixed set of resources of reconfigurable devices, on which NIDS is built. Using a single recognition method leads to the situation that some resources (for example, logical) are involved almost completely, while others (resources of block or external memory) are not used at all. As a result, efficiency is lost.

D. STRUCTURE OF THE METHOD

The Fig. 2 schematically presents the structure of the PCM.

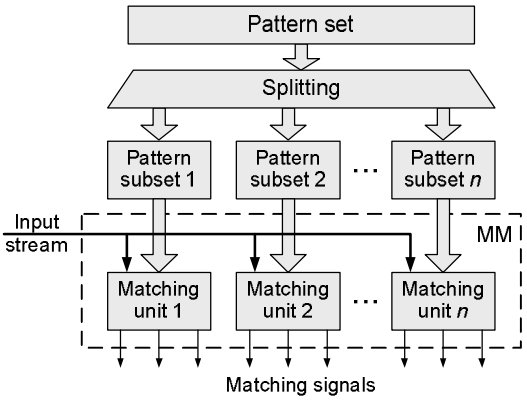


Fig. 2. The structure of the Parallel Combination Method. Schematic representation

During synthesis, the pattern set P to be recognized is divided into n pattern subsets P_i, by the splitting procedure, $i = 1 \ldots n$. When functioning, each MU in the MM searches for the patterns of the corresponding subset in the input data stream, which is fed simultaneously to the inputs of all MUs. If a fact of matching of input data fragment with some pattern is detected, the corresponding signal is activated.

To achieve maximum efficiency, the process of forming the MM structure is carried out under the control of the general optimization procedure.

Variable parameters are the number of pattern subsets n and a certain splitting combination of the subsets P_i, as well as certain combination of units MU_i, which are selected from the library of components.

Table 1

The comparison of the main approaches to the construction of the reconfigurable multi-pattern matching modules

No.	Parameter		Approach		
			CAM	Bloom Filter	Aho-Corasick
1.	Logic costs		- - -	+	+++
2.	Memory costs	distributed	- - -	+	+++
3.		BRAM	+++	+	- - -
4.		external	+++	+++	- - -
5.	Speed		+++	+	-
6.	Speed predictability		+++	- - -	+++
7.	Scalability	by bandwidth	+++	+	-
8.		by pattern set size	- - -	+++	- - -
9.		by pattern length	-	+++	+++
10.	Ability to use redundancy of pattern set		+	- - -	+++
11.	Functional parameters	dynamic update	- - -	+	+++
12.		the ability to counter attacks targeted at the defense system	+++	- - -	+
13.		ability to work in NIPS mode	+++	+	-
14.	A significant drawback that negates the main advantages of the approach		Excessive resource costs	Fixed pattern length	"Explosive" memory growth

Notation: "+++" – significant advantage; "+" – medium advantage; "- - -" – significant drawback; "-" – medium drawback.

The optimization criteria when using PCM can be chosen depending on the needs of the user of NIDS. This question is discussed in the next section.

E. OPTIMIZATION PROCEDURE

Different optimization criteria can be used depending on the NIDS user's needs. In any case, the goal of optimization is to minimize or maximize a certain objective function, which is the numerical value of a certain technical parameter: consumed resources, performance, productivity, and so on.

To gain the goal of optimization the algorithm that implements PCM varies the variable parameters, calculating the resource and time characteristics of MU_i and entire MM at each step.

Of course, the characteristics of the reconfigurable matching module can be found by synthesizing its digital circuit using the design tool from FPGA developer. But this process takes too much time [31] that it makes such an approach unfeasible for PCM implementation.

Therefore, the author proposed a technique of the accelerated calculation of the characteristics of MU_i and MM. Its essence is to create the so-called estimation function (EF) for each library component. Such a function, having a given set of patterns P_i at the input, calculates the amount of resources (and/or time characteristics) that the *i*th MU has to consume to be able to recognize this pattern subset.

F. ESTIMATION FUNCTIONS

Due to the distinction of the nature of approaches, estimation functions for them are being composed in different manner. For example, because of transparent and regular structure of CAM-based matching unit circuits, the EF for it can be formed by direct calculation of the required resources.

Let us consider in detail how such EF is formed using the example of a basic scheme of pattern matching BsCAM based on discrete comparators [28], clarifying along the way some important points about building DC-based CAM on FPGA.

The BsCAM scheme implements directly the function of detecting the matching of the input word with the pattern and consists of a set of discrete comparators working in parallel, each of which compares the input byte with a predefined symbol [12], [13]. The Fig. 3 shows such a circuit, which contains a pipeline consisting of 8-bit registers RG_i, comparators CMP_1 … CMP_3, each of which performs the function of comparison with a certain symbol, and a logical circuit "AND", which aggregates their outputs. The set of comparators corresponds to one pattern to be recognized. The analyzed sequence of symbols is fed to the input of the pipeline. In case a fragment of sequence matches the pattern "ABC", the active signal appears on the output "Match".

A scheme in the Fig. 3 looks pretty simple. But some difficulties rise in its practical implementing on FPGA.

Since the signature database of modern security systems can contain a large number of patterns [26] he output load ability (fan-out) of registers RG_i, which are made from conventional FPGA components, becomes insufficient. And a high length of the digital lines connecting a large number of logic elements distributed along the surface of the FPGA chip leads to delays in signal propagation, which reduces the maximum operation frequency of the entire digital circuit.

To solve the problem, a pipeline of several stages can be created [32]. Each output signal is branched into a plurality of inputs of D-latch of the next stage (Fig. 4).

This solution at the expense of moderate additional hardware costs and acceptable increase in latency allows to distribute the outputs of the RG_i registers to any number of comparators without reducing the clock frequency.

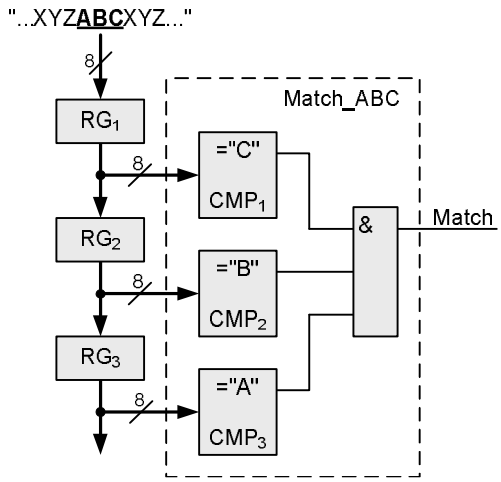


Fig. 3. A basic scheme of direct matching the "ABC" pattern by discrete comparators

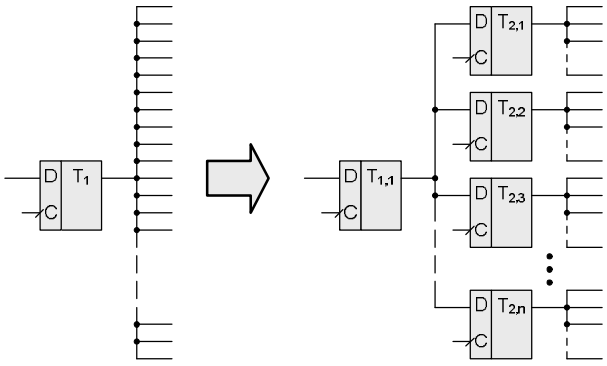


Fig. 4. Solving the FAN-OUT problem

Another problem of comparators-based scheme is related to the large number of inputs of logic circuit "AND". The length of pattern can be of tens of characters, whereas the number of inputs of lookup table (LUT) of modern FPGAs, which implement this scheme, does not exceed 8. This difficulty can also be solved by pipelining (Fig. 5) [12] [32].

We can now return to composing the estimation function for the BsCAM scheme.

In general case, the resources required for synthesis in the FPGA of any MU (as well as the whole MM in general) can be calculated in conventional units which are equivalent in terms of costs, for example, to one LUT:

$$R_i = L_i + \alpha F_i + \beta B_i + \gamma M_i, \quad (1)$$

where L_i – amount of resources of logics of FPGA, which is required to synthesize ith unit (number of LUTs); F_i – amount of resources of distributed memory of FPGA (number of flip-flops), B_i – amount of resources of block memory of FPGA (number of BRAM blocks), M_i – amount of external memory resources – on-board memory of reconfigurable accelerator (Mb), α, β, γ – normalization coefficients of different type resources in relation to logic resources (LUTs).

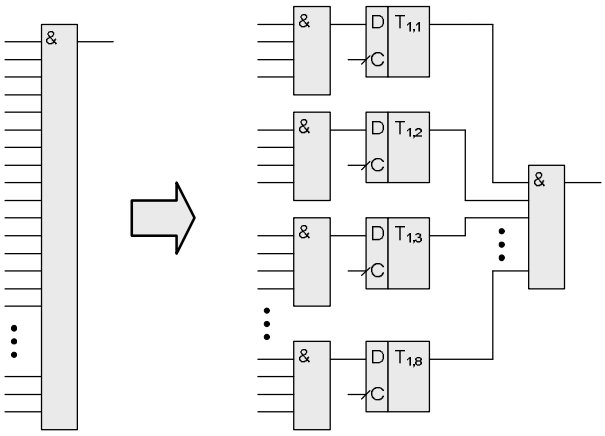


Fig. 5. Solving the problem of multi-input "AND" circuit

Because the logical cells of most modern FPGAs have the same number of LUTs and flip-flops ($\alpha = 1$), and because the CAM-based matching schemes do not require BRAM or external memory, the expression (1) for the basic scheme on discrete comparators BsCAM is simplified to:

$$R_{\text{BSCAM}} = L_{\text{BSCAM}} + F_{\text{BSCAM}}, \quad (2)$$

where L_{BSCAM} and F_{BSCAM} – numbers of LUT and flip-flops in this scheme respectively.

The lookup tables LUT are used in the scheme BsCAM, firstly, for synthesizing comparators CMP (Fig. 3), secondly – to create multi-input circuit "AND" (Fig. 5), which is built as a pipeline [28]:

$$L_{\text{BSCAM}} = L_{\text{CMP}} + L_{\&}. \quad (3)$$

The feature of the BsCAM scheme is the fact that it requires as many comparators Ω as there are characters in all the patterns of the set to be recognized:

$$\Omega = \sum_{j=m_{\min}}^{m_{\max}} \delta_j \cdot j, \quad (4)$$

where j – length of a pattern in the set; $m_{\min}$ – length of the shortest pattern; $m_{\max}$ – length of the longest pattern; δ_j – pattern length distribution function.

The discrete comparator CMP, which recognizes one character in byte encoding, requires two 4- or 6-input LUTs, or one LUT having 8 (or more) inputs, when constructing CAM schemes. Let us introduce the qualifier function

$$\Lambda(x) = \begin{cases} 1, & x \geq 8 \\ 2, & x < 8 \end{cases},$$

where x – the number of inputs of the LUT for a given FPGA. Then the number of LUTs required for creating all comparators in BsCAM scheme will be equal to

$$L_{\text{CMP}} = \Lambda(x) \cdot \Omega. \quad (5)$$

Taking into account the fact that the x-input LUT can synthesize a logic circuit "AND" not more than for x inputs, the number of LUTs required to join j inputs with a cascade circuit "AND" (Fig. 5) for each pattern is equal [33] to:

$$L_j = L(j) = \left\lceil \frac{j-1}{x-1} \right\rceil. \quad (6)$$

The number of LUTs to form all pipelines of all patterns is calculated similarly to (4) take into account (6):

$$L_{\&} = \sum_{j=m_{\min}}^{m_{\max}} \delta_j \cdot \left\lceil \frac{j-1}{x-1} \right\rceil. \quad (7)$$

The number of flip-flops required to create a BsCAM scheme consists of their number F_{RG} required to build the input pipeline, the number $F_{\text{fan-out}}$ of flip-flops to increase the FAN-OUT of the input pipeline registers (Fig. 4) and the number $F_{\&}$ of flip-flops in the pipeline, which joins comparator outputs for all patterns by function "AND" (Fig. 5):

$$F_{\text{BSCAM}} = F_{\text{RG}} + F_{\text{fan-out}} + F_{\&}. \quad (8)$$

The pipeline, on which the input stream of characters moves, is not shorter than the length of the longest pattern of subset $m_{\max}$ and has one byte width. Therefore, the number of flip-flops required to build it is as follows:

$$F_{\text{RG}} = 8 \cdot m_{\max}. \quad (9)$$

Finding the number $F_{\text{fan-out}}$ (Fig. 4) is more difficult compared to the previous calculations. The load on the output registers of the input pipeline (Fig. 3) is irregular.

The signals from its first $m_{\min}$ stages are used to recognize all the patterns of the subset whereas the signals from the last stage – only to recognize the longest patterns. By performing all the necessary calculations [28], we get:

$$F_{\text{fan-out}} = m_{\min}\left\lceil \frac{\sigma-1}{y-1}\right\rceil + \sum_{i=m_{\min}+1}^{m_{\max}}\left\lceil \frac{\sum_{i=j}^{m_{\max}}\delta_i - 1}{y-1}\right\rceil, \quad (10)$$

where σ – number of patterns in the subset; y – FAN-OUT property of the given FPGA.

The number of flip-flops in the pipeline for combining "AND" for each pattern is less for one than the number of LUTs in this pipeline $L_{\&}$ according to (7), because a flip-flop is not required after the last stage of the scheme "AND" (Fig. 5):

$$F_{\&} = L_{\&} - 1 = \sum_{j=m_{\min}}^{m_{\max}} \delta_j \cdot \left\lceil \frac{j-1}{x-1}\right\rceil - 1. \quad (11)$$

Substituting (5) and (7) into (3), as well as (9), (10) and (11) into (8), and then – (3) and (8) into (2), and taking into account that the flip-flops and LUTs, which are involved in the synthesis of the pipeline circuit of the multi-input circuit "AND", can be used together in the same logical cell of FPGA, we obtain the total number of computing resources of the DC-based CAM basic scheme BsCAM:

$$R_{BSCAM} = \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left(\Lambda(x) j + \left\lceil \frac{j-1}{x-1}\right\rceil\right) + $$
$$+ 8m_{\max} + m_{\min}\left\lceil \frac{\sigma-1}{y-1}\right\rceil + \sum_{i=m_{\min}+1}^{m_{\max}}\left\lceil \frac{\sum_{i=j}^{m_{\max}}\delta_i - 1}{y-1}\right\rceil. \quad (12)$$

Let us analyze the result obtained. The function (12) which estimates resources for Matching module built using the basic BsCAM scheme on digital comparators depends, on the one hand, on the parameters of the pattern set: σ, $m_{\min}$, $m_{\max}$, δ, on the other – on the characteristics of the FPGA chip used in the reconfigurable accelerator: x and y. When performing the optimization procedure (splitting into subsets), the parameters of the pattern set are variables, while the characteristics of the FPGA are constants.

The above example for the BsCAM scheme allows to get a certain understanding of the process of estimate function composition. Without extra details, let us look at the EF representation for a simplified pattern matching scheme BFS based on a Bloom filter [29], and for Aho-Corasick finite automaton using block memory ACBRAM [30]

The EF for the BFS scheme is as follows [34]:

$$R_{\text{BFs}} = \left\lceil \frac{e}{p}\right\rceil \cdot \left(\alpha \cdot G + \beta + \left\lceil \frac{G}{\left\lfloor \frac{x-1}{2}\right\rfloor}\right\rceil + 4\right) +$$
$$+ e \cdot G \cdot \left(\left\lceil \frac{8L}{x}\right\rceil + (\alpha+1)\cdot \left\lceil \frac{\left\lceil \frac{8L}{x}\right\rceil - 1}{x-1}\right\rceil - \alpha\right) + \left\lceil \frac{\left\lceil \frac{e}{p}\right\rceil - 1}{x-1}\right\rceil, \quad (13)$$

where e – recognition error factor, which is numerically equal to the number of HFs in the BF (is inversely proportional to the logarithm of the probability of false positive, which is acceptable for a particular application of the Bloom filter); p – number of ports of block memory BRAM;

$$G = \left\lceil \log_2 \frac{e \cdot \delta_L}{\ln 2}\right\rceil$$

– bit width of the hash-function generators,

Where δ_j – pattern length distribution function; L – length of the patterns that are recognized by this BF.

The estimation function for the ACBRAM scheme has such appearance as [34]:

$$R_{\text{ACBRAM}} = L_{\text{CU}} + r\left(\left\lceil \frac{\lceil \log_2 r\rceil}{x}\right\rceil + 1 + \beta\right) + w \cdot \left\lceil \frac{r-1}{x-1}\right\rceil +$$
$$+ \alpha\left(F_{\text{CU}} + w \cdot \left(\left\lceil \frac{r-1}{x-1}\right\rceil - 1\right) + \lfloor \log_2 (M_{\text{BRAM}}/w)\rfloor\right),$$

where L_{CU} and F_{CU} – number of LUTs and flip-flops required to create a control device for AC FA respectively, which can be found by a certain approximating technique; M_{BRAM} – amount of memory (in Kbits) in one BRAM block of the given FPGA (excluding parity bits, if any); $r = \left\lceil \frac{B}{1024(M_{\text{BRAM}}/w)}\right\rceil$ – number of BRAM block required, where B – amount of block memory required for the ACBRAM scheme (in Mbits), which is the sum of the amounts of memory for direct, cross, failure and post-start transitions of the AC FA: $B = B_{\text{dr}} + B_{\text{cr}} + B_{\text{fl}} + B_{\text{ps}}$ [30]; w –width (in bits) of data stored in BRAM (depending on the technique used to build the finite automaton).

V. RESOURCE COSTS OPTIMIZATION

Optimization task formulated in section IV has high complexity. To solve it for the appropriate time, some heuristic algorithms can be applied that allow us to find an approximate optimum, not global, but at moderate time costs.

One of them is to sort all patterns in some order and split them into two subsets simply placing the first several items to one subset, and the rest – to another one. Obviously, such algorithm is applicable only when only two approaches are involved.

Let us try to invent some heuristics to split patterns.

By examining the table in section III, one can make such an observation. A significant drawback of CAM that negates its main advantage is the excessive consumption of resources. While for BF it is a fixed length of patterns.

That is, for patterns of different lengths you need to build a separate Bloom filter. On the other hand, the CAM-based approach has very poor scalability by the pattern set size whereas BF-based one – vice versa.

Hence, if we split the pattern set so that the MU built on the BF will recognize as many patterns with the same length as possible, and the MU built on the CAM – as few patterns as possible, the advantages of both approaches will be mutually reinforcing.

So, let us group all the patterns into packages of the same length and sort them by the number of patterns in the package. Firstly, we will give all patterns to the scheme on CAM, and then consequently take away the packages with the greatest numbers of repetitions to the BF-scheme: the biggest package, then the biggest and the second by size package (leaving the rest of them to the CAM-scheme), then – the first three packages, etc. In the end, all packages (i.e. all the patterns) will be recognized by the scheme on BF, while the CAM scheme will have nothing to process. At each step of the algorithm we calculate the hardware costs for each MU and the total value using the EFs.

The Fig. 6 shows the resource costs calculated in conventional units according to the algorithm formulated above. The estimation function (12) of the BsCAM scheme was used for the CAM-based MU simulating, and the expression (13) of BFS scheme – for the BF-based MU simulating. In this experiment, pattern set from the signature database *community.rules* of free NIDS Snort ver. 06/23/19 was used. The downward curve indicates resource costs of BsCAM scheme. The upward curve – of BFS scheme.

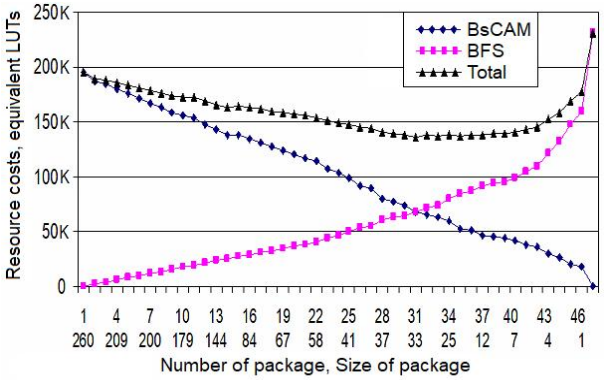


Fig. 6. Resource costs of the combined scheme

The leftmost point is the case when all the patterns are matched by BsCAM device. In the first step one package of 260 patterns of equal length are being recognized by BFS, the rest of patterns – by BsCAM. In the second step two packages of 260 and 259 equal patterns are processed by BFS, the rest – by BsCAM. In the third step three packages of 260, 259 and 243 patterns are processed by BFS. In the fourth step – four packages of 260, 259, 243 and 209 patterns are processed by BFS, the rest of patterns are matched by BsCAM. And so on.

The rightmost point corresponds to the situation when all patterns are matched by BFS device.

The upper curve indicates resource costs of the whole MM including both BsCAM and BFS MUs. As it can be seen, the total value curve has a pronounced minimum near mark of 35th packet. This means that if we split the pattern set so that the BF-based MU process the first 35 packets (having majority patterns of the same length) and the CAM-based MU process the rest of set (predominantly different patterns), the combined Matching module will consume about 30 % fewer resource costs than "pure" BsCAM scheme or about 50 % fewer than the BFS one. Thus, combining two approaches together reduces the cost of resources compared to using one approach.

VI. CONCLUSION AND FUTURE WORK

The most well-known approaches to the construction of reconfigurable information protection tools are: content addressable memory based on digital comparators; Bloom filter based on hash-functions; the Aho-Corasick algorithm implemented in the form of a finite automaton. Numerous researchers have also considered a lot of modifications and improvements to the basic solutions. But none of these approaches has a significant advantage over others in terms of parameters of efficiency.

The contributions of this work are as follows.

Specific features of different approaches in terms of resource costs, speed/throughput parameters, functional characteristics, as well as scaling parameters are formulated and explored. The universal technique to increase the effectiveness of FPGA-based hardware by combining different multi-pattern string matching approaches and particular technical solutions was regarded. A method of combining in parallel several matching units based on different approaches was constructed. The speeding-up technique based on calculating estimation functions for every approach was considered and discovered. An example of patterns splitting algorithm was calculated. The obtained results show the significant benefits of the proposed method and allow developers to construct more effective signature-based reconfigurable devices for security applications.

In future it is planned to investigate another than parallel methods of combining different approaches to build reconfigurable pattern matching devices.

VII. ACKNOWLEDGMENTS

This work was partially supported by the Program of Informatization of the NAS of Ukraine for 2015-2019.

REFERENCES

[1] ChenH., ChenY. , and Summerville D. H., "A Survey on the Application of FPGAs for Network Infrastructure Security", *IEEE Communications Surveys and Tutorials*, Article vol. 13, no. 4, pp. 541–561, 2011, doi: 10.1109/surv.2011.072210. 00075.

[2] Hilhurt S. Y., "Application of FPGA-based reconfigurable accelerators for network security tasks", *Modeling and*

information technologies. Collection of scientific works of PIMEE NAS of Ukraine, no. 73, pp. 17–26, 2014.

[3] Paxson V. *et al.*, "Rethinking hardware support for network analysis and intrusion prevention", presented at the USENIX First Workshop on Hot Topics in Security (HotSec), Vancouver, July 31, 2006.

[4] Smyth B., *Computing Patterns in Strings*. Essex: Pearson Addison Wesley, 2003.

[5] Lewis H. R. and Papadimitriou C. H., *Elements of the Theory of the Computations*, 2nd ed. Prentice-Hall, 1998.

[6] Kazmirchuk S., Korchenko A., and Paraschuk T., "Analysis of intrusion detection systems", Ukrainian Information Security Research Journal, vol. 20, no. 4, pp. 259–276, 2018, (in Ukrainian), doi: 10.18372/2410-7840.20.13425.

[7] Korostil J. M. and Hilgurt S. Y., "Principles of building FPGA-based network intrusion detection systems", *Modeling and information technologies. Collection of scientific works of PIMEE NAS of Ukraine*, no. 57, pp. 87–94, 2010, (in Russian).

[8] Katashita T., Yamaguchi Y., Maeda A. and Toda K., "FPGA-based intrusion detection system for 10 Gigabit Ethernet", *IEICE Transactions on Information and Systems*, Article vol. E90D, no. 12, pp. 1923–1931, Dec 2007, doi: 10.1093/ietisy/e90-d.12.1923.

[9] Hilgurt S., "Constructing optimal reconfigurable pattern matching tools for information security", *Ukrainian Scientific Journal of Information Security*, vol. 25, no. 2, pp. 74–81, 2019, (in Ukrainian), doi: 10.18372/2225-5036.25.13824.

[10] Guccione S. A., Levi D., and Downs D., "A reconfigurable content addressable memory", *Parallel and Distributed Processing, Proceedings*, Article; Proceedings Paper vol. 1800, pp. 882–889, 2000.

[11] Cho Y. H., Navab S., and Mangione-Smith W. H., "Specialized hardware for deep network packet filtering", *Field-Programmable Logic and Applications, Proceedings: Reconfigurable Computing Is Going Mainstream*, Article; Proceedings Paper vol. 2438, pp. 452–461, 2002.

[12] Sourdis I. and Pnevmatikatos D., "Fast, large-scale string match for a 10Gbps FPGA-based network Intrusion Detection System", *Field-Programmable Logic and Applications, Proceedings*, Article; Proceedings Paper vol. 2778, pp. 880–889, 2003.

[13] Y. H. Cho and W. H. Mangione-Smith, "Deep packet filter with dedicated logic and read only memories", *12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, Proceedings*, Proceedings Paper pp. 125–134, 2004, doi: 10.1109/fccm.2004.25.

[14] Sourdis I. and Pnevmatikatos D., "Pre-decoded CAMs for efficient and high-speed NIDS pattern matching", *12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, Proceedings, Proceedings Paper pp. 258–267, 2004, doi: 10.1109/fccm.2004.46.

[15] Sourdis I., Pnevmatikatos D. N. and Vassiliadis S., "Scalable multigigabit pattern matching for packet inspection", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Article vol. 16, no. 2, pp. 156–166, Feb 2008, doi: 10.1109/tvls1.2007.912036.

[16] Bloom B. H., "Space/Time Trade-offs in Hash Coding with Allowable Errors", *Communications of the ACM*, Article vol. 13, no. 7, pp. 422–426, 1970, doi: 10.1145/362686.362692.

[17] Fan L., Cao P., Almeida J., and Broder A. Z., "Summary cache: A scalable wide-area Web cache sharing protocol", *IEEE - ACM Transactions on Networking*, Article vol. 8, no. 3, pp. 281–293, Jun 2000, doi: 10.1109/90.851975.

[18] Dharmapurikar S., Krishnamurthy P., Sproull T. S. and Lockwood J. W., "Deep packet inspection using parallel bloom filters", *IEEE Micro*, Article; Proceedings Paper vol. 24, no. 1, pp. 52–61, Jan-Feb 2004, doi: 10.1109/mm.2004.1268997.

[19] Dharmapurikar S., Attig M. and Lockwood J., "Design and Implementation of a String Matching System for Network Intrusion Detection using FPGA-based Bloom Filters", in *All Computer Science and Engineering Research, Washington University in St. Louis, WUCSE-2004-12*, 2004-03-25 2004.

[20] J. Harwayne-Gidansky, D. Stefan, and I. Dalal, "FPGA-based SoC for Real-Time Network Intrusion Detection using Counting Bloom Filters", presented at the *Proceedings of the IEEE SoutheastCon 2009*, Technical Proceedings, 2009, Proceedings Paper.

[21] Geravand S. and Ahmadi M., "Bloom filter applications in network security: A state-of-the-art survey", *Computer Networks*, Article vol. 57, no. 18, pp. 4047–4064, Dec 2013, doi: 10.1016/j.comnet.2013.09.003.

[22] Aho A. V. and Corasick M. J., "Efficient String Matching: An Aid to Bibliographic Search", *Communications of the ACM*, vol. 18, no. 6, pp. 333–340, 1975, doi: 10.1145/360825.360855.

[23] Lunteren J., "High-performance pattern-matching for intrusion detection", *25th IEEE International Conference on Computer Communications, Vols 1-7, Proceedings IEEE Infocom 2006*, Proceedings Paper pp. 1409-1421, 2006.

[24] Lin C., Y. Tai, and Chang S., "Optimization of pattern matching algorithm for memory based architecture", presented at the *Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems*, Orlando, Florida, USA, 2007.

[25] Pao D., Lin W. and Liu B., "Pipelined architecture for multi-string matching", *IEEE Computer Architecture Letters*, vol. 7, no. 2, pp. 33–36, 2008, doi: 10.1109/L-CA.2008.5.

[26] Jiang W., Yang Y. H. E. and Prasanna V. K., "Scalable multi-pipeline architecture for high performance multi-pattern string matching", in *24th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2010*, Atlanta, GA, 2010, doi: 10.1109/IPDPS.2010.5470374.

[27] Lin C. H. and Chang S. C., "Efficient Pattern Matching Algorithm for Memory Architecture", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Article vol. 19, no. 1, pp. 33–41, Jan 2011, doi: 10.1109/tvlsi.2009. 2028346.

[28] Hilgurt S. Y., "Constructing CAM on discrete comparators by reconfigurable means for solving information security tasks", *Electronic Modeling*, vol. 41, no. 3, pp. 59–80, 2019, (in Ukrainian), doi: 10.15407/emodel.41.03.059.

[29] Hilgurt S., "Constructing Bloom filters by reconfigurable means for solving information security tasks", *Ukrainian Scientific Journal of Information Security*, vol. 35, no. 1, pp. 53–58, 2019, (in Ukrainian), doi: 10.18372/2225-5036.25.13594.

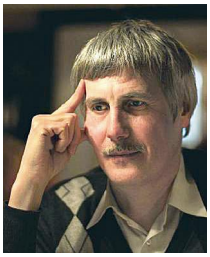
[30] Hilgurt S., "Constructing deterministic finite automata by reconfigurable means forsolving information security Tasks", *Ukrainian Information Security Research Journal*, vol. 21, no. 2, pp. 111–120, 2019, (in Ukrainian), doi: 10.18372/2410-7840.21.13768.

[31] Evdokimov V. F., Davydenko A. M. and Hilgurt S. Y., "Organization of centralized generation of bitstreams for hardware accelerators for information security tasks", *Modeling and information technologies. Collection of scientific works of PIMEE NAS of Ukraine*, no. 81, pp. 3–11, 2017, (in Russian).

[32] Huang J., Yang Z. K., Du X., and Liu W., "FPGA based high speed and low area cost pattern matching", in *IEEE Region 10 Conference (TENCON 2005),* Melbourne, AUSTRALIA, Nov 21-24 2005, NEW YORK: IEEE, 2006, pp. 2693–2697.

[33] Hilgurt S. Y., Kislov O. G., Popova V. M. and Liakh I. M., "Accelerated calculation of characteristics of reconfigurable matching schemes based on CAM and digital comparators", *Modeling and information technologies. Collection of scientific works of PIMEE NAS of Ukraine*, no. 89, pp. 3–16, 2019, (in Ukrainian).

[34] Evdokimov V. F., Davydenko A. M., Hilgurt S. Y. and Yarema O. R., "Hardware platform for reconfigurable information security tools", *Modeling and information technologies. Collection of scientific works of PIMEE NAS of Ukraine*, no. 86, pp. 3–11, 2019, (in Ukrainian), doi: 10.5281/zenodo.610626.



Sergii Hilgurt, a Senior Researcher of Pukhov Institute for Modelling in Energy Engineering (PIMEE), NAS of Ukraine, since 1992. He received the scientific degree PhD in "Computing machines, complexes, systems and networks" from the graduate school of the Institute for Modelling in Energy Engineering, Kyiv, Ukraine, in 1990.

From 1994 to 2008, he worked part-time at oil-pipelines automation company GERAX.

From 2000 to 2004, he studied at the Doctorate of PIMEE. He was awarded the scientific title of Senior Researcher in "Computer systems and parts", in 2015. He is the author of two books, two preprints, five inventions, and more than 80 articles. His research interests include: factory automation, computing hardware, HPC, reconfigurable acceleration, DPI, multi-pattern string matching, FPGA-based network security systems and information security of critical cyber-physical systems.

Mr. Hilgurt has been the Administrator of the GRID-site UA-PIMEE (UNG/EGI) and MatModEn Virtual Organization, since 2008.