A Survey on Hardware Solutions for Signature-Based Security Systems

Serhii Ya. Hilgurt[a]

[a] *Pukhov Institute for Modelling in Energy Engineering of NAS of Ukraine, 15, General Naumov str., Kyiv, 03164, Ukraine*

Abstract

Signature approach due to its exact matching is still a relevant technology used in applications like network intrusion detection systems (NIDS), antivirus, anti-spam, worm-containment and other security systems. Basing on signature approach, multi-pattern string matching technology by comparing an input data stream against a set of predefined patterns, detects malicious activities. Due to the increasing signature database size and the high throughput of today's networks the traditional software alone solutions can no longer meet the performance requirements of such systems. Therefore, many hardware approaches are proposed to accelerate the computational-intensive pattern matching procedure. On the other hand, a developer of hardware-accelerated security system has to solve non-trivial problem of choosing the most suitable approach in each particular case. Thus, some kind of guidance is needed to facilitate this problem solving. This paper provides a comprehensive survey on the hardware solutions were proposed in this area for the past years focusing on their effectiveness and feasibility. A deep analysis of main directions of security systems hardware acceleration has done. The most promising direction based on the use of FPGA and programmable logic is more closely investigated.

Keywords[1]

Security, signature, multi-pattern matching, NIDS, FPGA, CAM, Bloom filter, Aho–Corasick

1. Introduction

Whereas the propagation of Internet and network technologies in all spheres of human activities gains in many benefits the increase in number and sophistication of attacks against the network infrastructure and computer systems significantly escalates risks of information security compromising. Therefore, many technologies and techniques are offered to make information systems more robust and secure.

Despite the great progress made in recent years by AI-based approaches, in particular, by deep neural networking, security tools based on such methods still suffer from nonzero recognition error probability. Even low probability of such error is able to disrupt the correct operation of the security information system. In critical infrastructure such effects can have disastrous consequences. Therefore, signature-based recognition methods with their theoretically exact match are still relevant when creating information security systems.

Signature-based technology relies on using so called signatures – descriptions of known attacks that are compared against the intensive flow of input information (coded by bytes). Checking every byte of every data element to see if it matches one of hundreds thousand and millions strings becomes a computationally-intensive task as network speeds grow into the tens and hundreds of gigabits/second. Thus, the multi-pattern string matching task has been a major performance bottleneck in information

ITTAP'2021: 1st International Workshop on Information Technologies: Theoretical and Applied Problems, November 16–18, 2021, Ternopil, Ukraine
EMAIL: hilgurt@ukr.net (S.Ya. Hilgurt)
ORCID: 0000-0003-1647-1790 (S.Ya. Hilgurt)

security systems which have to scan the incoming data in real time [1]. There is an ever-widening performance gap between the security systems processing requirements and their software implementations because of the speed limitations of sequential software execution on standard CPUs [2]. To keep up with these speeds specialized devices built on hardware-oriented solutions are required.

Literature review demonstrates the high variety of technical solutions aimed to the hardware acceleration of multi-pattern string matching based on different computing platforms, which use approaches to building matching schemes of very different nature. Thus, choosing a solution to be most suitable for particular application is a complex and cunning problem that developers of signature-based security systems aim to solve effectively.

This paper provides some kind of guidance for such developers through analyzing, systematizing and classifying known platforms, approaches, techniques and customized solutions are invented by researchers over the world to realize the multi-pattern matching task in hardware. The goal of this work is also to facilitate the process of creating signature-based hardware-accelerated security systems by becoming familiar with their features and peculiarities.

To continue it is necessary to decide on some terms and definitions.

2. Terms and definitions

Some of the definitions required to work with signature-based information security tools are as follows.

The *signature* in this paper is considered as aggregate information about a specific threat, stored in the signature database of the correspondent security system.

The term *pattern* refers to a sample of a text string (a fixed sequence of characters in a certain encoding), which is part of the signature and is to be find in the input data.

Pattern dictionary – all the patterns that the database of signatures contains.

Pattern set – patterns that only some signatures contain that are selected for a certain purpose.

The mathematical formulation of the *task of multi-pattern string matching* can be defined as follows. D. Gusfield in the monograph [3], page 53, defines it as generalization of the task of exact matching: "An immediate and important generalization of the exact matching problem is to find all occurrences in text T of any pattern in a set of patterns $P = \{P_1, P_2, \ldots, P_Z\}$".

So, given text T that consists of written in the line n symbols: "$C_1C_2C_3 \ldots C_n$", and the set of patterns $P = \{P_1, P_2, \ldots, P_Z\}$, where Z is its size, and each pattern P_i ($i = 1, 2, \ldots, Z$) consists of written in the line m symbols: "$C_1C_2C_3 \ldots C_m$", where m is the length of the corresponding pattern (different for each of them in the General case), the task of multi-pattern string matching is to identify all occurrences in the text T any of the patterns of R_i from the set P.

The fundamental difference multi-pattern string matching task from the single string recognition, which is effectively solved by the classical algorithms Knuth–Morris–Pratt, Boyer–Moore, Karp–Rabin, etc., is in the need to search not one pattern, but many patterns simultaneously.

In a practical sense, the task of multi-pattern matching is to provide answers to the following questions:

- whether there are substrings in the text T, that matches character by character with any of the patterns P_i;
- if any – with which ones;
- how many matches for each of the patterns is detected;
- in which position (relative to the beginning of the text T) is each of the matching substrings.

Note that the sequences of characters corresponding to the patterns in the input data may not only be repeated, but also partially overlap, i.e. the end of the suspicious substring may match the beginning of another. Such situations must also be revealed when solving the multi-pattern matching task [4].

3. HPC vs. accelerators

Traditional way to sole computationally-intensive problems is use of high performance computing means as supercomputers, clusters, GRID and cloud. Unfortunately, the main feature of the entire HPC segment is the apartness of their computing resources from the place where the calculation results are needed. The large volume and intensity of information to be processed when performing multi-pattern matching do not allow the use of the above-mentioned high-performance solutions. In other words, the local nature of security tasks in fact makes it impossible to apply HPC equipment.

Instead, it turned out to be more acceptable to use various joined coprocessors (accelerators), which are located directly at the place of the problem, so do not require quick transfer of large amounts of data over long distances. The principles of construction and capabilities of such devices vary considerably. But the main feature that unites their diversity is the use of hardware solutions instead of software, because the main purpose of such devices is to overcome the limitations faced by software solutions, as mentioned above.

4. Hardware platforms for multi-pattern matching

Devices that theoretically can be used as a hardware platform for solving the multi-pattern matching task include:

- specialized coprocessors based on ASIC;
- network processors (NP);
- ternary content-addressable memory (TCAM);
- accelerators based on multi-core processors;
- graphics accelerators (GPGPU);
- based on FPGA reconfigurable accelerators (RA).

Let's consider the features of these technologies and their possibilities to solve the multi-pattern matching task for further comparison.

4.1. Specialized coprocessors based on ASIC

In specialized hardware coprocessors, due to the constriction of the target area, the maximum acceleration is achieved the microelectronic technology can theoretically provide at present. The ASICs (Application-Specific Integrated Circuits) is the base of modern special processors that are in fact the top of the semiconductor circuit evolution, starting with small- and medium-scale integration devices, whose electrical circuits have been redesigned for each application up to complex nanometer process products, which allow developers to synthesize the most complex computing schemes.

Creation of special hardware devices for solving the information security tasks on the basis of ASIC chips is quite possible, and such developments exist. Due to specialization, such solutions have the best performance and energy consumption characteristics compared to other technologies. But the production of each ASIC is a resource-intensive process both in terms of cost and time. The fixed internal structure of such integrated circuits does not allow making changes in already made products. Properly speaking, a special processors based on ASIC is not a joined accelerator in the conventional sense, because products on "fixed" logic a-priory cannot be universal. Therefore, the use of special ASIC-based processors is reasonable under the condition of sufficiently large production volumes.

4.2. Network processors

Network processors were created to accelerate the execution of specialized network tasks, including multi-pattern matching for signature-based information security systems. A typical NP in addition to the components of a conventional network adapter contains a CPU, usually of VLIW- or RISC-architecture, multi-port memory and additional logic resources to perform typical network operations. On such a device it is convenient to organize computing, for instance, according to the multi-pattern matching algorithm Aho–Corasick by using additional logic for a finite automaton control unit, the memory – to store transition table and the CPU – to create and load these tables, and to perform general control functions [5].

There are also NPs that includes content-addressable memory chips with three states – the ternary content-addressable memory (see the next subsection).

But despite the fact that NPs were designed specifically for network problems, the fixed architecture and limited computing resources have led to a decline in interest for this type of accelerators, and they have not actually been used now.

4.3. TCAM-memory

The tertiary content-addressable memory, strictly speaking, is not a separate computing platform [6], [7]. But the speed capabilities of TCAM (response time in nanosecond range) are of some interest and distinguish it from other hardware that focuses on the algorithmic performance of multi-pattern matching. A number of methods have been developed to use TCAM for multiple matching, but without addressing its practical implementation as part of an accelerator.

In contrast to conventional binary-, ternary-memory provides limited possibility for flexible matching based on regular expression [6]. There are also several comparative studies in which TCAM is considered as a computing technology, so this type of equipment was moved in this paper to a separate section. Disadvantages of the approach based on using TCAM chips are: a fixed architecture and a limit on the pattern dictionary volume.

4.4. Multi-core processors

A typical multi-core device is the Intel Xeon Phi product [8]. The main advantage of this technology is ease of programming. The similarity to conventional processors allows using well known and verified algorithmic approaches for multi-pattern matching. Strong computing capabilities of multi-core processors simplify the implementation of complex procedures of multi-pattern matching, such as flexible recognition using regular expressions [9].

Disadvantages include the well-known shortcomings of microprocessor technology – fixed internal structure and increased power consumption, as well as difficulties in organizing data exchange with the central (control) computer and between processor cores, that complicates the parallelization of multi-pattern matching task.

4.5. Graphic technology GPGPU

The technology of using graphics accelerators not for the main purpose, but to perform resource-intensive computing was called GPGPU (General-Purpose Graphics Processing Units).

The process of displaying graphical information on the screen of a personal computer initially was more complex than the actual data processing in the CPU. And this task with the development of the PC only continued to get more complicated. The market for graphics adapters was equal to market PCs, so manufacturers have invested heavily in the development of graphics processing units. The most powerful products in terms of functionality and performance contained an increasing number of components capable to process information independently. In order to unify technical solutions and simplify their management processes, well-known low-level programming techniques were used. As a result it became unexpectedly possible not only to send data to the GPU for processing, but also to retrieve results of data processing back.

Modern GPUs consist of many independent processing elements, capable to simultaneously process a large number of data streams [10]. Processing elements can be programmed independently or similar to vector architecture. That is, GPGPUs can function both as SIMD (Single Instruction Multiple Data) architecture and as a SPMD (Signal Program Multiple Data) structure, which is a kind of the MIMD (Multiple Instruction, Multiple Data) architecture class. Like multi-core processors, GPGPUs realize mainly algorithmic methods for solving multi-pattern matching tasks.

Disadvantages of GPGPU technology also include a fixed internal structure (moreover optimized for graphics problems) and excessive power consumption. The advantages of this platform are: high overall performance and fine-tuned interaction with the central computer.

Therefore, GPGPU platform can be effectively used to solve the multi-pattern matching task.

4.6. FPGA-based reconfigurable accelerators

The most flexible and universal tool for creating digital circuits, which allows developing computing devices for any purpose at almost any level of complexity is programmable logic [11]. The ability of modern FPGAs (which already contain millions of equivalent gates) to synthesize information processing devices by any principles, approaches and computational architectures gives the developer a wide space to choose and implement the most appropriate direction for creating recognition tools. Namely the possibility to implement the latest methods and algorithms for information processing in digital devices should be considered a significant advantage of programmable logic.

Because the computing scheme created in FPGA is in fact a specialized device, it does not consume excess electrical power to support universal components, which makes RAs a more energy saving option compared to other platforms.

The use of RA as joined accelerator allows building high-performance computing system on the basis of universal computer.

The presence of a large number of ready-made products that can be used as RA is an advantage of reconfigurable platform too. An electronic resource [12], numbering more than a thousand items, makes it possible to estimate the number and variety of reconfigurable accelerators. Among the most well-known manufacturers of RAs are: Nallatech, Xilinx, Alpha Data, National Instruments, DiNI Group and many others.

However the main advantage of the direction based on the FPGAs and RAs should be considered the ability to quickly (in a few seconds) change the functionality of the security device.

The disadvantages of the programmable logic platform include the relative high cost and complexity of digital schemes synthesis and FPGA configuration files generation.

5. Hardware platforms comparison

As mentioned above, network processors and ternary content-addressable memory chips have recently lost interest from developers and are no longer used. These platforms were considered only for completeness of the study. Thus, for comparison, ASIC, GPGPU, Multi-Core and FPGA technologies remain.

The problem of choosing the most suitable hardware platform for signature-based security system is complicated. A rigorous and well-grounded solution would be possible if for each platform there were examples of experimental development of the same task in the same formulation, detailed to the quantitative level. Unfortunately, no information about such developments has been found in scientific publications and practical reports. In the best case, two or three platforms were compared, and furthermore, one of them often was a traditional solution on the CPU. Moreover, the comparison was mainly carried out qualitatively rather than quantitatively.

Using the available sources of all types, as well as other accessible information and logical considerations, let's conduct with the maximum degree of objectivity a comparative analysis of the above hardware platforms for solving signature-based security tasks.

One of the only works where almost all technologies are compared [7] provides a qualitative assessment of the parameters of physical constraints, system design and performance (Table 1). The study [13] qualitatively presents advantages and disadvantages of Multi-Core, ASIC and FPGA technologies (Table 2). Quantitative comparison of certain developments on Multi-Core, GPGPU, ASIC and FPGA platforms can be found in [14]. Some data from this publication are given in Table 3.

Note that in the Table 3 the clock speeds of Multi-Core, GPGPU and ASIC projects are significantly exceed the frequencies of FPGA-based solutions. This indicates the high potential of the reconfigurable platform, for which, unlike other areas, the technological limit has not yet been reached, and i.e. the so-called Moore's Law still works for the FPGA platform.

Indeed, in one of more recent sources [15], it is reported that the corresponding implementation of a specific signature-based network security task based on FPGA, according to the results of a number of experiments, outperformed a traditional processor by 24-89 times, a 12-core processor by 4.7-7.4 times. Comparison of the reconfigurable solution with the implementation of the same task on the

GPGPU platform with 200 parallel streams showed an advantage of FPGA in bandwidth by 3.14 times, propagation delay by 1020 times, and power consumption by 106 times.

Table 1
Implementation alternatives for signature matching

Parameters		ASIC	FPGA	GPU	NP	CPU
Physical constraints	Cost	Highest	Medium	Low	Medium-Low	Low
	Power efficiency	Highest	Low-Medium	High	Medium	Lowest
	Area efficiency	Highest	Worst	High	Medium	Low
	Scalability	High	Low	Medium	High	High
System design	Flexibility	Worst	Medium	Medium	Medium	Best
	Design time	Highest	Medium	Low	Low	Lowest
Performance	Peak performance	Highest	Medium	Medium	Medium	Lowest
	Application performance	Highest	Medium	High-Medium	Medium	Lowest

Table 2
Comparison among hardware platform solutions

Platform	Multi-Core Processors	ASIC	FPGA
Advantage	Can enhance the aggregate throughput dramatically by using a large number of threads to process multiple input stream in parallel	Provide impressively high per-stream throughput	Provide desirable high performance, flexibility of software and reconfigurable programming
Disadvantage	Additional Complexity is introduced in scheduling, buffering ordering, and load balancing	The applicability is limited by the high implementation cost and low reprogrammability	It takes considerable time to resynthesize the design and reprogram the FPGA device

Table 3
Quantitative comparison

Approach	Platform	Num. of Pattern	Patterns length	Clock rate (MHz)	Throughput (Gbps)
AC -DFA	Multi-Core (Cell/B.E.)	8400	≤10	3200	2.5
AC -DFA	GPGPU (GeForce 8600GT)	4000	≤25	1200	2.3
CDFA	ASIC (0.18 µm)	1785	No limit	763	6.1
Bit-Split	FPGA	1316	No limit	220 (200)	1.76
B-FSM	FPGA	~8000	No limit	138 (125)	2.2
Field-Merge	FPGA	6944	< 64	285	4.56
Multi-Pipeline	FPGA	9033	No limit	178	11.4

In summary, we reasonably conclude that FPGA-based reconfigurable accelerators as a hardware platform best meet the complex and dynamic nature of information security tasks, outperforming competing technologies in terms of flexibility and cost-effectiveness. Therefore, the rest of the paper is devoted to a more detailed consideration of this technology and a comparison of the approaches used when constructing multi-pattern matching schemes on reconfigurable accelerators.

6. FPGA-based reconfigurable accelerators

Programmable logic integrated circuits have been used successfully in engineering for a long time.

The form of implementation of the FPGA-based equipment as joined co-processors is a reconfigurable accelerator (Figure 1).

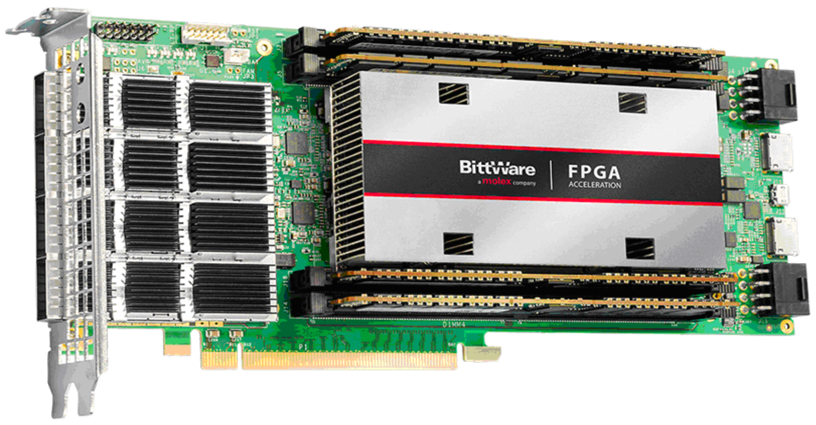

Figure 1: Appearance of a typical reconfigurable accelerator

The structure of such a device contains both mandatory and additional components [16]. Every RA contains at least one FPGA chip. Another mandatory component of the RA is onboard random access memory (RAM) for storing intermediate results of calculations, which are usually used standard chips of large capacity DDR-memory.

A separate hardware interface controller is also highly recommended to be an RA component. In addition to communicating with the host system's central processor, this controller also provides FPGA chip programming, i.e. loads the configuration file (bitstream) into it.

An important additional component of RA is the terminal equipment of communication channels for direct high-speed exchange with the external environment, bypassing the CPU of the host system.

Today, the world produces a large number of FPGA-based devices that can be used as RA and implemented for solving information security tasks. These devices can range from high-performance, general-purpose reconfigurable accelerators to less functional but more affordable products such as starter kits, trainer boards, evaluation kits, development boards and prototype plates. The Table 4 exemplifies several reconfigurable accelerators, which cover a wide range of devices in terms of specifications and functionality.

Due to the relatively long obsolescence of RAs, accelerators with very different capabilities are used concurrently. As we can see, some parameters of RA devices in the table vary by several orders of magnitude.

7. Constructing reconfigurable multi-pattern match circuits

Many different approaches to constructing hardware matching circuits on FPGAs were proposed recently. Hundreds researches publish thousands of papers devoted to this scientific direction. To understand this diversity and correctly assess the possibilities of various solutions, it is necessary to set the appropriate efficiency criteria. Before deciding on the indicators, let us consider the most prevalent class of protection tools - network intrusion detection systems as an example of a signature-based security system.

Table 4
Specifications for typical RAs

Manufacturer	Nallatech	Nallatech	Xilinx	Digilent	Digilent
Model of RA	XUP-VV8	385A	VC709 Kit	NetFPGA-1G-CML	Atlys
Kind of RA	Full-size RA	Full-size RA	Evaluation Kit	Development Board	Trainer Board
Type of FPGA	Xilinx Virtex UltraScale+ VU13P	Intel (Altera) Arria 10 GX 1150	Xilinx Virtex-7 VX690T	Xilinx Kintex-7 XC7K325T	Xilinx Spartan-6 LX45
Number of logical cells	3 780 000	1 150 000	693 120	326 080	43 661
BRAM size, Kbit	36	20	36	36	18
Total volume of BRAM memory, Mbit	94,5	54,3	52,9	16,0	2,1
Type of onboard RAM	DDR4 SDRAM	DDR3 SDRAM	DDR3 SDRAM	DDR3 SDRAM	DDR2 SDRAM
Onboard RAM volume, GB	512	32	8	0,512	0,128
Network ports	4x QSFP-DD (2x 100 Gb - Ethern. each)	2x QSFP28 (100 Gb - Ethern. each)	4x SFP+ (10 Gb - Ethern. each)	4x RJ-45 PHY (1 Gb - Ethern. each)	1x RJ-45 PHY (1 Gb - Ethern.)
Communication interface	PCI-Express x16 Gen.3	PCI-Express x8 Gen.3	PCI-Express x8 Gen.3	PCI-Express x16 Gen.2	USB 2.0
Form factor	Dual-slot PCI-E board standard-height 3/4-length	Single-slot PCI-E board half-height half-length	Single-slot PCI-E card standard-height/ length	Single-slot PCI-E card standard-height 3/4-length	stand-alone board 120 x 133 mm
Cooling	Passive	Active	Passive	No	Passive
Price, $	8995	5995	4995	1499	490

7.1. The structure of reconfigurable NIDS

NIDS were historically the first and, consequently, the most studied FPGA-based tools of information security [17]. Therefore, without losing the generality of reasoning, consider the typical functions and efficiency parameters of reconfigurable security systems on the example of such systems. The structure and composition of the reconfigurable network intrusion detection system (Figure 2) can be compiled as a generalization of numerous published scientific solutions (see for example [17], [18]).

The Matching module is the most important component of NIDS. It solves the computationally complex task of multi-pattern string matching, i.e. checks the content of network packets against pattern set. As we can see the signatures is "wired" into the circuitry of the device at hardware level. This feature ensures the highest performance rate due to the maximally possible parallelism has been reached.

7.2. The efficiency parameters of reconfigurable NIDS

There are three main groups of efficiency parameters (or indicators) of reconfigurable NIDS [19], [20]: cost parameters, parameters of speed or performance parameters, functional parameters.

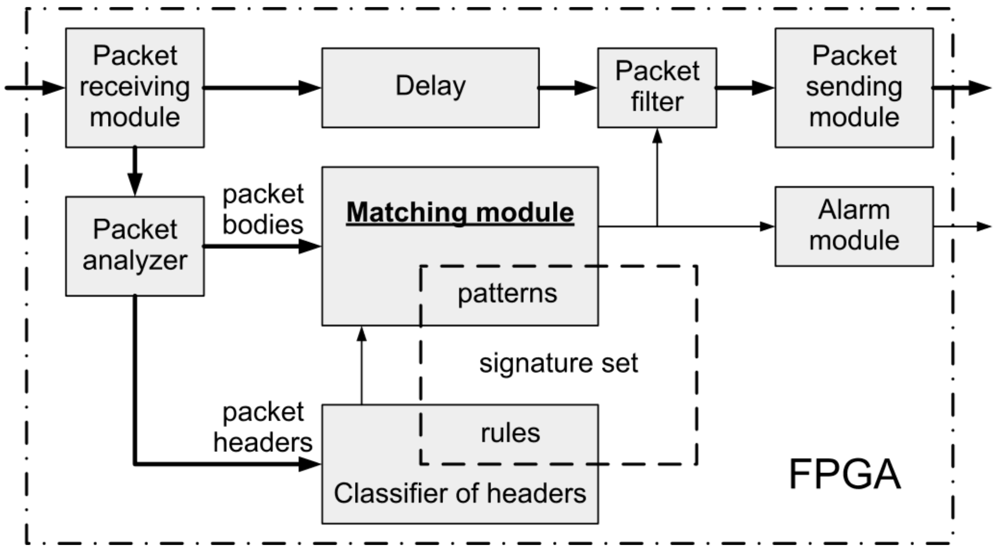

Figure 2: The structure of the reconfigurable network intrusion detection system

Cost indicators are as follows:

- the amount of logical resources of programmable logic needed to create a digital circuit;
- memory costs of three types: onboard memory of RA, internal memory of FPGA (BRAM) and distributed memory of FPGA (flip-flops in logical cells);
- other costs, which make up the total cost of ownership, including the development, manufacture and programming costs.

Performance parameters include:

- the volume of the pattern dictionary (i.e. the number of patterns to be matched);
- the speed of the system (which is defines as either the delay of data propagation from input to output or as bandwidth);
- the predictability of bandwidth as well.

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;
- ability to selective recognition (the ability to change a subset of recognizable patterns by an external signal);
- the ability to counter attacks targeted at the NIDS itself,

and others.

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.

7.3. Comparison of the main approaches to the building of reconfiguring matching modules

The analysis of numerical researches from all over the world shows that when creating NIDS the most effective and widely applied are these three approaches which use the following technical solutions based on the corresponding technologies:

- content-addressable memory (CAM) based on digital comparators (DC) [21], [22], [23], [24], [25] and [26];
- Bloom filter (BF) based on hash-functions (HF) [27], [28], [29], [30], [31] and [32];
- Aho–Corasick algorithm (AC) on finite automata (FA) [14], [33], [34], [35], [36], [37] and [38].

Each of these approaches has its own pros and cons. And none of these fully meets the requirements for the reconfigurable signature-based security systems. The lack of a leading direction that would outperform competitive solutions in all respects makes the developers offer numerous modifications of the main approaches, use diverse techniques and technical solutions trying to overcome their shortcomings.

Let's consider briefly these technologies, approaches and its modifications, techniques and solutions for every of three mentioned directions.

7.3.1. Approach based on content-addressable memory and digital comparators

Content-addressable memory is a class of devices that were created for quick code recognition and perform a function opposite to traditional RAM: it finds the location of data in the memory device by its content or reports about their absence. Digital comparators are the fast-acting basis of CAM [39].

The direct solution for detecting the matching of input symbols with patterns is a set of digital comparators, each of which compares the input byte with a predetermined symbol [23], [24]. Let's call such scheme the *Basic scheme on CAM* or the *BsCAM* scheme (Figure 3). Just the input data fed, the BsCAM circuit is able to give the output in one clock cycle. The simplicity and regularity of the structure are also its advantages. The main disadvantage is the significant consumption of logical resources of FPGA.

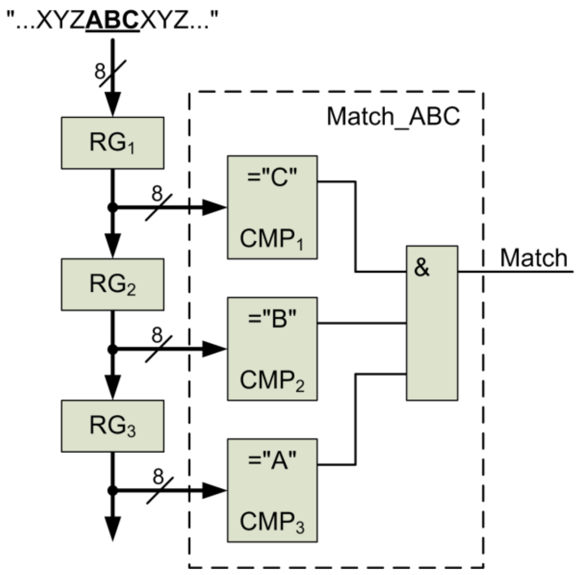

Figure 3: Direct recognition by digital comparators

Difficulties when implementing on FPGA arise for the following reasons. Firstly, the outputs of registers built on standard FPGA components are overloaded by a large number of comparators inputs. Secondly, it is necessary to combine a large number of bit signals with a multi-input AND gate for long patterns. Both problems are solved by pipelining the fan-out schemes, which leads to an additional increase in hardware costs [23], [25]. This leads to the poor scalability of the approach by the pattern set size.

It is possible to reduce the CAM hardware costs of by the reuse of comparators, which in the extreme case leads to the allocation of only one DC per character of the alphabet, and their combinations are formed using delay circuits and the AND gates, as shown in Figure 4. Here DEL(1) and DEL(2) – delay circuits of one and two cycles, respectively. This solution is called the *Decoded Content-Addressable Memory* (DCAM), because the complete set of the DCs forms a decoder [25].

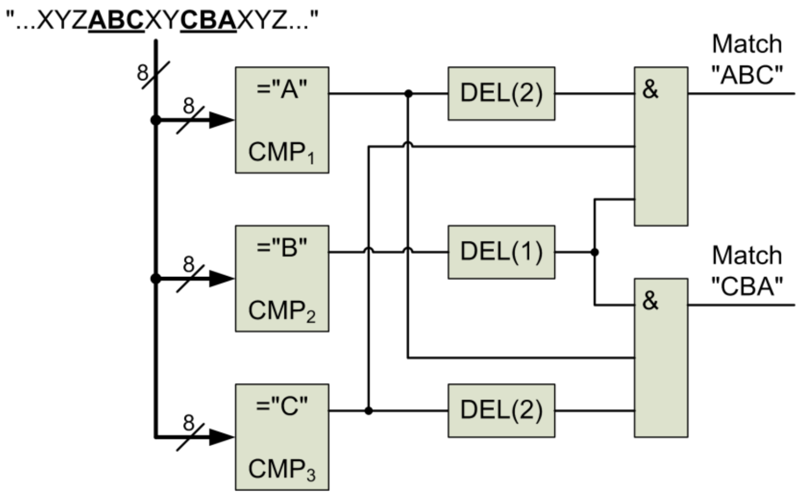

Figure 4: DCAM schematic solution

A further reducing the complexity of the DCAM-based recognition scheme is possible by using a partial matching technique where long patterns are broken into shorter parts that are recognized consistently. It is sufficient to delay only the partial match signal instead of using long delay circuits for a large number of cycles for long patterns. Figure 5 depicts the recognition scheme of a 31-character pattern, built on this principle [26]. This solution was named the *Recognition Scheme based on partially decoded CAM* or the *DpCAM* (Decoded partially Content-Addressable Memory) scheme.

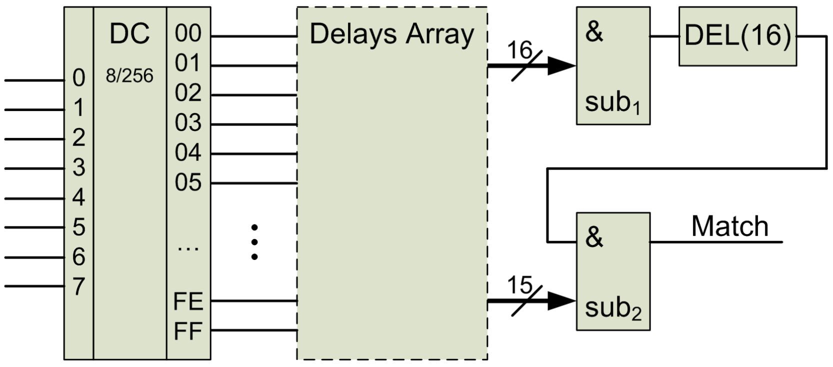

Figure 5: DpCAM schematic solution

A more detailed study shows that the use of the cost reduction techniques described above in the case of their parallel connection (in the *ParCAM* scheme) leads to even higher resource savings than linear growth [23]. Thus, the schemes on the DC have a good scalability by the bandwidth.

There is also a technique for reducing costs by using non-byte data processing, which is implemented by *Hbc*, *HbcDCAM* and *ParHbcDCAM* schemes on half-byte comparators [40], as well as the *BCAM* scheme, which is built using binary decision diagram [41].

The fact that the input data content does not affect the operation of the DC-based CAM recognition scheme implements the bandwidth predictability functional efficiency indicator and makes security system invulnerable to external attacks, which is another functional efficiency indicator.

As can be seen from the considered solutions, when creating recognition tools on DCs, the information about the patterns is actually "wired" into the hardware circuit, which makes it impossible to dynamically reconfigure and fulfill the selective recognition.

The results obtained by examining the features of the recognition schemes based on CAM and DC are summarized in Table 5 below (see the section 7.4. "Comparison of the approaches").

7.3.2. Approach based on Bloom filter and hash-functions

The Bloom filter is an abstract device that allows detecting a match of a fragment of a given bit sequence with a sample from the dictionary [27]. BF consists of two key components (Figure 6): a set of K units that calculate the hash functions $h_1(x)$, $h_2(x)$, …, $h_K(x)$, and an array of M one-bit memory cells (component Rg in the figure). In the initial state, this bit register (BR) is filled with zeros [31].

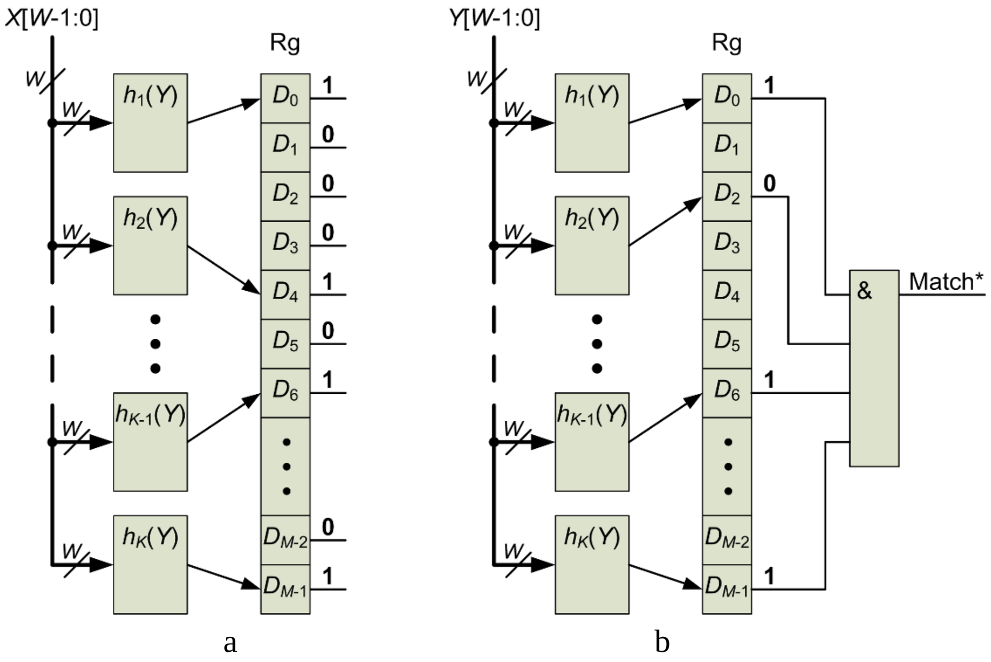

Figure 6: The principle of operation of the Bloom filter: a – programming; b – recognition

At the programming stage of the Bloom filter, each of the n elements of the pattern dictionary (W-bit length) is sequentially fed to the inputs of the hash function units. The output values are interpreted as addresses (bit numbers) of BR and correspondent cells are filled with ones.

While functioning, a fragment of the input stream of symbols (also of W bits length) is fed to BF input, and the values of all K hash-functions are calculated as well. If all the cells of BR pointed by the hash-functions outputs contain ones, it is assumed with certain probability that the input combination of characters matches one of the patterns used while the BF programming process. But if at least one hash-function points to a zero value, no match situation is guaranteed. Thus, BF operates with some probability of false positive error, but without false negative errors [32].

BF saves memory resources. The size of the pattern set does not directly affect the size of the BR: adding new patterns to existing ones only increases the probability of a recognition error, but does not increase the amount of memory required. The number and complexity of hash-functions, and therefore hardware consumption and performance, also do not depend on the volume of the pattern set with this approach. The pattern length also does not affect directly the amount of memory resources or performance. That is, BF scales well in two ways: by pattern set size and by pattern length. Indeed, even very long strings after conversion with hash-functions require the same K cells of BR to be stored.

Unfortunately, the Bloom filter has an important drawback inherent in all hash-based solutions: the size of the input character sequence to be analyzed must exactly correspond to the given set of hash-functions. That is, one BF is able to recognize patterns of only the same length. To overcome this shortcoming, it is necessary to build a structure containing several BF in order patterns of different lengths can be used [29]. This reduces the cost-effectiveness of the Bloom filter approach, but not the speed indicators.

Difficulties when implementing BF on FPGA arise primarily because several hash-function generators need to simultaneously access the bit register Rg (Figure 6). When implementing the latter on a single memory device, collisions will occur. The solution is to split the BR into several memory devices depending on the number of hash-functions. This requires, firstly, distributing the outputs of the hash-function units between the corresponding RAM devises, and secondly, imposing additional restrictions on the functionality of these units, so that output range of every unit fit the reduced size of memory device. It has been proven that such constraints not significantly increase the false positive error probability of Bloom filter [29].

In the general case, several BRAM blocks are required to create a BR. The corresponding circuit is called a *Full-Size Bloom Filter* or *LBF* (Large Bloom Filter). But in most practical applications for security system the so-called mini-BF circuit, which are part of the LBF and contain only one block of BRAM, can be used for the matching module construction. Let's call such decision the *Simplified Bloom filter* or the *SBF* (Simplified Bloom Filter) scheme [30].

In order to increase the speed, BF can be combined into a parallel structure similar to the ParCAM scheme. But unlike DC, the properties of Bloom filter do not allow to achieve a sublinear law of cost growth – the hardware costs of a parallel circuit are strictly proportional to the achieved acceleration, i.e. BF bandwidth is not scaled as well as CAM-based circuits.

The classic scheme of the Bloom filter does not allow on-the-run removing patterns added during programming. If necessary, developer can use a Bloom filter with counters, or *CBF* (Counting Bloom Filter) scheme [28]. This solution allows dynamic reconfiguration without stopping the security system.

The need to fulfill the procedure of refining the results due to BF's system error, which is slower than recognition, leads to unpredictability of bandwidth, and makes the Bloom filter vulnerable to attacks on security systems [30], [31].

The results of examining properties of the recognition schemes based on BF are also summarized in Table 5.

7.3.3. Approach based on Aho–Corasick algorithm and finite automata

The mathematical apparatus of finite-state machines or finite automata (FA) has been used successfully to create computer systems for a long time [42]. A special class of FAs – so-called *classifiers* is widely used when building signature-based security systems that perform multi-pattern string matching [43]. Such a machine outputs an active signal only if one of the predefined sequences of symbols is received at its input. When this situation occurs, the classifier goes into one of the so-called *acceptable* states [1]. While transitions are performed between unacceptable states, there are no signals at the output of a FA–classifier.

The Aho–Corasick algorithm is an example of a tool that, unlike many known single string recognition algorithms, detects several samples in the input data simultaneously [33]. The essence of the algorithm is in creating an FA according by certain rules, which recognizes the desired patterns while operating. Let's call such FA the *Aho–Corasick finite automaton* (AC-FA).

The theory and practical experience of using AC-FA is widely presented in the literature [14], [34], [35], [36], [37] and [38]. However, for the effective building AC-FA with help of FPGAs, special attention needs to be paid to the finite automaton transition function. Analysis of the problem gives an understanding of the need to distinguish following four types of transitions in such machines: direct or goto transitions, cross, failure and restartable transitions. The latter type, strictly speaking, belongs to cross transitions, but its separation into an individual type allows increasing the effectiveness of FPGA-based AC-FA realization due to some peculiarities of technical implementation of such machines [44].

A generalized structure of the hardware implementation of AC-FA is shown in Figure 7. It based on a memory unit (MU) in which the machine transition table is stored [36], [38]. Each MU cell contains a number of the next state of automaton and a match vector (MV). A character code from the input stream is concatenated with the value of the next state number retrieved from memory unit. The obtained value is fed to the MU address input and selects the appropriate record. MVs contain ones in the positions denoting the corresponding pattern. The control unit (CU) controls the operation of the machine. If the RA onboard memory (external to the FPGA chip) is used when implementing the circuit, we will call this solution the *Basic AC-FA Scheme with External Memory* or *ACRAM* scheme. A solution based on BRAMs (FPGA internal memory blocks) will be called Basic AC-FA Scheme with Block Memory or *ACBRAM* scheme.

The most important advantage of AC-FA solutions is its bandwidth independence from the signature database size and the pattern's properties, in particular, their lengths, which results in predictability of bandwidth. Typically, the FA takes one character from the input stream for each clock cycle. But in practice, if the size of transition table is too large and it is necessary to use external memory, each access to the MU can take several cycles. Furthermore, external memory is slower than internal. Thus, the downside of this advantage is the relatively low speed, which is also difficult to increase, which means poor AC-FA scalability by the bandwidth.

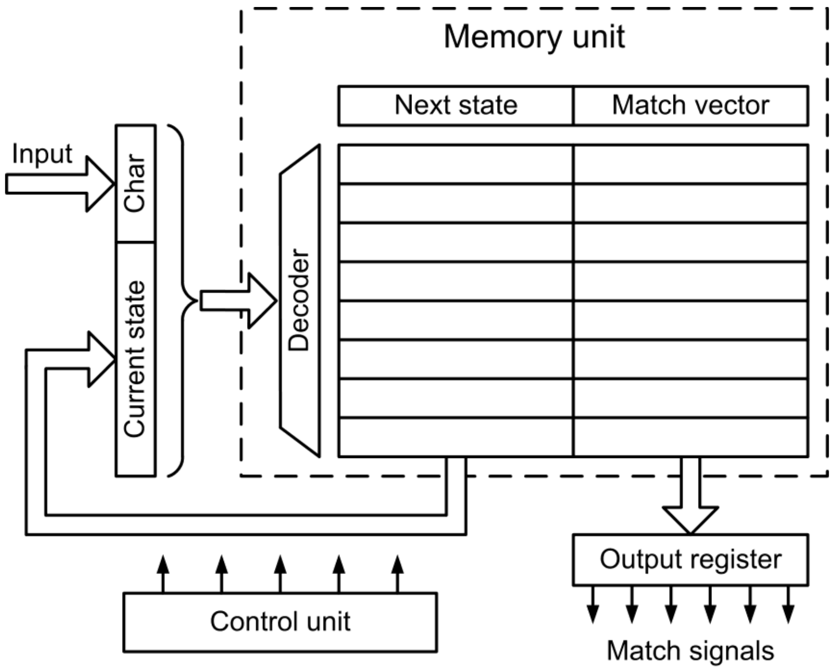

Figure 7: The structure of the typical Aho–Corasick finite automaton

The main difficulties that arise when implementing AC-FA on the reconfiguration platform concerns the organization of efficient data exchange with MU. When constructing an ACBRAM scheme there is a possibility to synthesize a memory device of almost arbitrary structure due to the flexibility of BRAM blocks configuring process. But in the case of ACRAM the problem is escalated because the onboard memory of RA has a fixed structure not convenient in the general case for interaction with CU [45].

As we can see, AC-FA requires a small amount of logic resources to create a control unit, registers and MU controller. However, the size of memory can be very large, i.e. the approach is characterized by high memory consumption. Increasing the number of patterns leads to "explosive" growth of MU resources, which means very poor scalability by pattern set size. Therefore, the majority of researchers' efforts when implementing AC-FA hardware in security systems are aimed at reducing memory resources and moderating their rapid growth.

Among the numerous modifications of AC-FA there are solutions that offer different ways to encode the transition table. The matches of not only prefixes but also infixes of patterns are also analyzed [36], [38]. However, in most modifications, memory reduction is achieved by handling certain types of finite automaton transitions [35], [44]. In a number of works the technique of AC-FA pipelining has been offered and developed [37]. The linear conveyor of H steps allows eliminating all the cross transitions in AC-FA structure from an initial state to level H [14].

A certain part of the researchers' efforts was aimed at improving the not very good speed efficiency of the approach. Because AC-FA, like any finite state machine, processes input information strictly sequentially, character by character, attempts have been made to speed up the AC algorithm by processing more than one character per clock cycle [46]. Since it is not known in advance with which offset the specific pattern will appear in the input data, it is necessary to organize the parallel operation of the appropriate number of identical machines (modifications *ParACRAM* and *ParACBRAM*).

Another solution related to non-byte data processing. The so-called *Bit-split* scheme was proposed to replace a FA that processes 8-bit characters by several identical parallel sub-machines that analyze 1, 2 or 4 bits [34]. By reducing the size of character codes, the size of the alphabet is significantly reduced. The Bit-split solution is the opposite of multi-character per clock cycle recognition schemes; however, it is not aimed at speeding up but at reducing resources.

The use of external RAM allows a simple implementation of the dynamic updating of the AC-FA circuitry by overwriting FPGA contents, i.e. to completely change the algorithm of the machine without stopping the security system. Regarding the indicator of selective recognition, it is implemented by forming in the MU several transition tables for different submachines.

The results of the study of FA-based recognition schemes properties that implement the AC algorithm are also summarized in Table 5.

7.4. Comparison of the approaches

The comparative analysis of three approaches: the use of associative memory based on digital comparators, Bloom filter based on hash functions and the Aho–Corasick algorithm based on finite automata, as well as numerous techniques for their improvement allows to thoroughly assessing the advantages and disadvantages of each of them in the sense of the performance indicators mentioned above. The Table 5 shows the results of this comparative analysis.

Table 5
The comparison of the main approaches to the construction of the reconfigurable multi-pattern matching modules

Parameter		Approach		
		CAM	Bloom Filter	Aho–Corasick
Logic costs		- - -	+	+++
Memory costs	distributed	- - -	+	+++
	BRAM	+++	+	- - -
	onboard	+++	+++	- - -
Speed		+++	+	-
Speed predictability		+++	- - -	+++
Functional parameters	the ability to counter attacks targeted at the defense system	+++	- - -	+
	dynamic update	- - -	+	+++
	selective recognition	- - -	+	+++
	ability to work in NIPS mode	+++	+	-
Scalability	by bandwidth	+++	+	-
	by pattern set size	- - -	+++	- - -
	by pattern length	-	+++	+++
Ability to use redundancy of pattern set		+	- - -	+++
A significant drawback that negates the main advantages of the approach		Excessive resource costs	Fixed pattern length	"Explosive" memory growth

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

As we can see, none of the researched approaches shows obvious advantages over others, each has its own positive features and disadvantages. And an advantage according to one of the indicators often turns into a disadvantage with respect to another. As a result, each of the approaches has a significant drawback, which, in fact, negates the main advantages of this approach.

For instance, CAM based on DCs and their modifications provide maximum performance, but more expensive than other approaches in terms of hardware resources and electricity consumption. They also lose in scaling. The Bloom filter is more scalable and effective by resources, but imposes restrictions on the length of the patterns. It also requires additional costs to check the obtained results due to its inherent false positive errors probability when matching. 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 pattern dictionaries.

8. Conclusion

Multi-pattern string matching is a fundamental technology used in signature-based security systems like network intrusion detection systems, antivirus, anti-spam, worm-containment and so on. This task is computational-intensive, and traditional software solutions do not keep up with speeds of modern networks because the number and sophistication of attacks against the computer infrastructure are increasing constantly. Hardware accelerators are able to overcome this bottleneck. There are several types of such devices known as acceptable solution: specialized coprocessors based on ASIC, ternary content addressable memory, accelerators based on multi-core processors, graphics accelerators and reconfigurable accelerators based on FPGA. The latter best meet the information security area requirements, outperforming competing platforms in flexibility and cost-effectiveness.

The most promising approaches to the construction of reconfigurable matching modules of signature-based security systems are: content addressable memory based on digital comparators, Bloom filter based on hash-functions and 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 efficiency.

The contribution of this work is as follows.

A comprehensive survey on the known hardware solutions in this area for the past few years has made. Most effective platforms, approaches, techniques and customized solutions to realize the multi-pattern matching task in hardware are analyzed, systematized and classified.

A comparative analysis of main hardware platforms for security systems acceleration has provided. The most promising direction using programmable logic is more closely explored.

Specific features of different approaches to the construction of FPGA-based matching schemes in terms of resource costs, speed/throughput parameters, functional characteristics, as well as scaling parameters are formulated and thoroughly investigated.

In fact, a developer guide aiming to facilitate choosing the most appropriate solution for a specific application of signature-based high performance security systems is proposed.

Acknowledgment

This work was supported in part by the Informatization Program of the National Academy of Sciences of Ukraine for 2020-2024.

References

[1] B. Smyth, Computing Patterns in Strings, Pearson Addison Wesley, Essex, 2003.

[2] H. Chen, Y. Chen, D. H. Summerville, A Survey on the Application of FPGAs for Network Infrastructure Security, IEEE Communications Surveys and Tutorials, Article volume 13, 4 (2011) 541-561. doi:10.1109/surv.2011.072210.00075.

[3] D. Gusfield, Algorithms on Strings, Trees, Sequences: Computer Science and Computational Biology, Cambridge University Press, Cambridge, 1997.

[4] T. AbuHmed, A. Mohaisen, D. Nyang, Deep Packet Inspection for Intrusion Detection Systems: A Survey, Computer Networks, volume 57, 18 (2012) 4047-4064.

[5] Y. N. Lin, Y. C. Chang, Y. D. Lin, Y. C. Lai, Resource allocation in network processors for network intrusion prevention systems, Journal of Systems and Software, Article volume 80, 7 (July 2007) 1030-1036. doi:10.1016/j.jss.2007.01.032.

[6] F. Yu, R. H. Katz, T. V. Lakshman, Gigabit rate packet pattern-matching using TCAM, in: Proceedings of the 12th IEEE International Conference on Network Protocols, 2004, pp. 174-183. doi:10.1109/icnp.2004.1348108.

[7] C. C. Xu, S. H. Chen, J. S. Su, S. M. Yiu, L. C. K. Hui, A Survey on Regular Expression Matching for Deep Packet Inspection: Applications, Algorithms, Hardware Platforms, IEEE Communications Surveys and Tutorials, Article volume 18, 4 (2016) 2991-3029. doi:10.1109/ comst.2016.2566669.

[8] Intel® Xeon Phi™ Processor 7290, 2016. URL: https://ark.intel.com/content/www/ru/ru/ark/ products/95830/intel-xeon-phi-processor-7290-16gb-1-50-ghz-72-core.html.

[9] Y. H. E. Yang, V. K. Prasanna, Optimizing regular expression matching with SR-NFA on multi-core systems, in: Proceedings of the Parallel Architectures and Compilation Techniques, PACT, 2011, pp. 424-433. doi:10.1109/PACT.2011.73.

[10] J. D. Owens et al., A survey of general-purpose computation on graphics hardware, in: Proceedings of the Computer Graphics Forum, Review volume 26, no. 1, 2007, pp. 80-113. doi:10.1111/j.1467-8659.2007. 01012.x.

[11] C. Maxfield, The Design Warrior's Guide to FPGAs: Devices, Tools and Flows, Elsevier Science & Technology Books, Oxford, UK, 2004.

[12] FPGA Boards and Systems, 2021. URL: http://www.fpga-faq.com/FPGA_Boards.shtml.

[13] R. Abdulhammed, M. Faezipour, K. M. Elleithy, Network Intrusion Detection Using Hardware Techniques: A Review, in: Proceedings of the IEEE Long Island Systems, Applications and Technology Conference (Lisat), 2016, p. 7.

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

[15] V. Jyothi, S. K. Addepalli, R. Karri, DPFEE: A High Performance Scalable Pre-Processor for Network Security Systems, IEEE Transactions on Multi-Scale Computing Systems, Article volume 4, 1 (January-March 2018) 55-68. doi:10.1109/tmscs.2017.2765324.

[16] S. Hilgurt, Reconfigurable accelerators. Analytical overview, Electronic modeling, Article volume 35, 4 (2013) 49-72, (in Russian).

[17] J. M. Korostil, S. Y. Hilgurt, Principles of building FPGA-based network intrusion detection systems, Modeling and information technologies. Collection of scientific works of PIMEE NAS of Ukraine, 57 (2010) 87-94, (in Russian).

[18] T. Katashita, Y. Yamaguchi, A. Maeda, K. Toda, FPGA-based intrusion detection system for 10 Gigabit Ethernet, IEICE Transactions on Information and Systems, Article volume E90D, 12 (December 2007) 1923-1931. doi:10.1093/ietisy/e90-d.12.1923.

[19] S. Hilgurt, Constructing optimal reconfigurable pattern matching tools for information security, Ukrainian Scientific Journal of Information Security, volume 25, 2 (2019) 74-81, (in Ukrainian). doi:10.18372/2225-5036.25.13824.

[20] S. Hilgurt, Parallel combining different approaches to multi-pattern matching for FPGA-based security systems, Advances in cyber-physical systems, volume 5, 1 (2020) 8-15.

[21] S. A. Guccione, D. Levi, D. Downs, A reconfigurable content addressable memory, in: Proceedings of the Parallel and Distributed Processing, vol. 1800, 2000, pp. 882-889.

[22] Y. H. Cho, S. Navab, W. H. Mangione-Smith, Specialized hardware for deep network packet filtering, in: Proceedings of the Field-Programmable Logic and Applications, Proceedings: Reconfigurable Computing Is Going Mainstream, volume 2438, 2002, pp. 452-461.

[23] I. Sourdis, D. Pnevmatikatos, Fast, large-scale string match for a 10Gbps FPGA-based network Intrusion Detection System, in: Proceedings of the Field-Programmable Logic and Applications, Proceedings: Reconfigurable Computing Is Going Mainstream, volume 2778, 2003, pp. 880-889.

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

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

[26] I. Sourdis, D. N. Pnevmatikatos, S. Vassiliadis, Scalable multigigabit pattern matching for packet inspection, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Article volume 16, 2 (February 2008) 156-166. doi:10.1109/tvlsi.2007.912036.

[27] B. H. Bloom, Space/Time Trade-offs in Hash Coding with Allowable Errors, Communications of the ACM, Article volume 13, 7 (1970) 422-426. doi:10.1145/362686.362692.

[28] L. Fan, P. Cao, J. Almeida, A. Z. Broder, Summary cache: A scalable wide-area Web cache sharing protocol, IEEE – ACM Transactions on Networking, Article volume 8, 3 (June 2000) 281-293. doi:10.1109/90.851975.

[29] S. Dharmapurikar, P. Krishnamurthy, T. S. Sproull, J. W. Lockwood, Deep packet inspection using parallel bloom filters, IEEE Micro, Proceedings Paper volume 24, 1 (January-February 2004) 52-61. doi:10.1109/mm.2004.1268997.

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

[31] J. Harwayne-Gidansky, D. Stefan, I. Dalal, FPGA-based SoC for Real-Time Network Intrusion Detection using Counting Bloom Filters, in: Proceedings of the IEEE SoutheastCon, 2009.

[32] S. Geravand, M. Ahmadi, Bloom filter applications in network security: A state-of-the-art survey, Computer Networks, Article volume 57, 18 (December 2013) 4047-4064. doi:10.1016/ j.comnet.2013.09.003.

[33] A. V. Aho, M. J. Corasick, Efficient String Matching: An Aid to Bibliographic Search, Communications of the ACM, vol. 18, 6 (1975) 333-340. doi:10.1145/360825.360855.

[34] L. Tan, T. Sherwood, A high throughput string matching architecture for intrusion detection and prevention, in 32nd International Symposium on Computer Architecture, Madison, WI, Jun 04-08 2005, LOS ALAMITOS: IEEE Computer Soc, 2005, pp. 112-122.

[35] J. Lunteren, High-performance pattern-matching for intrusion detection, in: Proceedings of the 25th IEEE International Conference on Computer Communications, volumes 1-7, 2006, pp. 1409-1421.

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

[37] D. Pao, W. Lin, B. Liu, Pipelined architecture for multi-string matching, IEEE Computer Architecture Letters, volume 7, 2 (2008) 33-36. doi:10.1109/L-CA.2008.5.

[38] C. H. Lin, S. C. Chang, Efficient Pattern Matching Algorithm for Memory Architecture, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Article volume 19, 1 (January 2011) 33-41. doi:10.1109/tvlsi.2009. 2028346.

[39] K. Pagiamtzis, A. Sheikholeslami, Content-addressable memory (CAM) circuits and architectures: A tutorial and survey, IEEE Journal of Solid-State Circuits, Article volume 41, 3 (March 2006) 712-727. doi:10.1109/jssc.2005.864128.

[40] J. Huang, Z. K. Yang, X. Du, W. Liu, FPGA based high speed and low area cost pattern matching, in: Proceedings of the IEEE Region 10 Conference (TENCON 2005), New York, 2006, pp. 2693-2697.

[41] S. Yusuf, W. Luk, Bitwise optimised CAM for network intrusion detection systems, in: Proceedings of the International Conference on Field Programmable Logic and Applications, FPL, Tampere, 2005, pp. 444-449. doi:10.1109/FPL.2005.1515762.

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

[43] R. M. Keller, Computer Science: Abstraction to Implementation. Harvey Mudd College, 2001.

[44] T. Song, W. Zhang, D. S. Wang, Y. B. Xue, A memory efficient multiple pattern matching architecture for network security, in: Proceedings of the 27th IEEE Conference on Computer Communications (Infocom), Volumes 1-5, 2008, pp. 673-681.

[45] Q. B. Wang, V. K. Prasanna, Multi-Core Architecture on FPGA for Large Dictionary String Matching, in: Proceedings of the 17th IEEE Symposium on Field Programmable Custom Computing Machines, 2009, pp. 96-103. doi:10.1109/fccm.2009.43.

[46] H. Lu, K. Zheng, B. Liu, X. Zhang, Y. H. Liu, A memory-efficient parallel string matching architecture for high-speed intrusion detection, IEEE Journal on Selected Areas, Communications, Article volume 24, 1 (October 2006) 1793-1804. doi:10.1109/jsac.2006.877221.