Hardware Approach for Generating b-detectors by Immune-Based Algorithms

Maciej Brzozowski, Andrzej Chmielewski

▶ **To cite this version:**

Maciej Brzozowski, Andrzej Chmielewski. Hardware Approach for Generating b-detectors by Immune-Based Algorithms. 13th IFIP International Conference on Computer Information Systems and Industrial Management (CISIM), Nov 2014, Ho Chi Minh City, Vietnam. pp.615-623, 10.1007/978-3-662-45237-0_56 . hal-01405656

HAL Id: hal-01405656

https://inria.hal.science/hal-01405656

Submitted on 30 Nov 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Hardware approach for generating *b*-detectors by immune-based algorithms

Maciej Brzozowski and Andrzej Chmielewski

Faculty of Computer Science, Białystok University of Technology,
ul. Wiejska 45a, 15-331 Białystok, Poland
{m.brzozowski,a.chmielewski}@pb.edu.pl

Abstract. The most interesting feature of negative selection algorithms is ability for detecting novel, never met anomalies. This is especially important in security systems like intrusion detection, spam, virus detection, etc. However, the main problem is scalability which occurs for both: binary and real-valued representation. This paper describes a hardware implementations of the process of generating *b*-detectors which allows for a fast generation the receptors as well as a very fast recognition of anomalies in high-dimensional datasets.

Keywords: artificial immune system, anomaly detection, hardware implementation

1 Introduction

Natural immune system (NIS) prevents living organism against intruders called *pathogens*. It consists of a number of cells, tissues, and organs that work together to protect the body. The main agents responsible for the adaptive and learning capabilities of the NIS are white blood cells called *lymphocytes*. These differentiate into two primary types: B- and T-lymphocytes called also B- and T-cells for brevity. T-lymphocytes are like the body's military intelligence system, seeking out their targets and sending defenses to lock onto them. Next, B-lymphocytes, destroys detected invaders to protected the body. It is only a very short description of NIS; an unacquainted reader is referred e.g. to [3] for further details.

The mechanisms and procedures developed within NIS were an inspiration for *Artificial Immune Systems* (AIS). One of them is *negative selection* oriented towards fast and efficient discrimination between own cells (called *self*) and pathogens (called *nonself*). Additionally, it helps the body against self-reactive lymphocytes. A nice feature is that it does not need examples of *nonself* samples (counterpart of pathogens) to detect them. The information about own cells is sufficient. Hence, every organism has a unique "protection system", capable of detecting even new type of attacks. This is an inspiration to build a computer security systems (e.g. firewall).

A key problem when applying *negative selection algorithm* (*NSA*) in real-life application is scalability. To detect spam or intruders at computer networks *NSA*

should be able operate on high-dimensional data. Moreover, in such domains, the classification is performed online, without significant delays, what makes this task much more difficult to solve. This descriptions correspond to NIS, where only fast and effective response on intruders activity can protect organisms against damaging or even die. To overcome this problem, many solution were proposed, including hybrid b-v model [4], emloying binary (called b-detectors) and real-valued detectors (called v-detectors) to increase the efficiency, especially in case of classification process.

In this paper, we present hardware implementations of *NSA* for b-detectors, where most of computationally complex operations can be paralellized and done within a few ticks of clock. This is the first step towards hardware firewall embedded in a reprogrammable FPGA (see Section 3). Such solution makes that rules in security systems are not hardcoded and updating the set of detectors is possible. Finally, there are compared two possible approaches: combinational and pipeline.

2 Negative Selection Algorithm

The NSA, proposed by Forrest *et al.*, [6], is inspired by the process of thymocytes (i.e. young T-lymphocytes) maturation: only those lymphocytes survive which do not recognize any *self* molecules.

Formally, let $\mathcal{U}$ be a universe, i.e. the set of all possible molecules. The subset $\mathcal{S}$ of $\mathcal{U}$ represents collection of all *self* molecules and its complement $\mathcal{N}$ in $\mathcal{U}$ represents all *nonself* molecules. Let $\mathfrak{D} \subset \mathcal{U}$ stands for a set of detectors and let $match(d, u)$ be a function (or a procedure) specifying if a detector $d \in \mathfrak{D}$ recognizes the molecule $u \in \mathcal{U}$. Usually, $match(d, u)$ is modeled by a distance metric or a similarity measure, i.e. we say that $match(d, u) = \texttt{true}$ only if $dist(d, u) \leq \delta$, where $dist$ is a distance and δ is a pre-specified threshold. Various matching function are discussed in [7], [10].

The problem relies upon construction the set $\mathfrak{D}$ in such a way that

$$match(d, u) = \begin{cases} \texttt{false} \text{ if } u \in \mathcal{S} \\ \texttt{true} \quad \text{if } u \in \mathcal{N} \end{cases} \tag{1}$$

for any detector $d \in \mathfrak{D}$.

A naive solution to this problem, implied by biological mechanism of negative selection, consists of five steps:

(a) Initialize $\mathfrak{D}$ as empty set, $\mathfrak{D} = \emptyset$.
(b) Generate randomly a detector d.
(c) If $math(d, s) = \texttt{false}$ for all $s \in \mathcal{S}$, add d to the set $\mathfrak{D}$.
(d) Repeat steps (b) and (c) until sufficient number of detectors will be generated.

So far, there were considered two types of detectors: b- and v-detectors [9]. Usually, each of them were used separately, except b-v model [4]. In this paper, we focus only on binary representation.

In case of binary encoding, the universe $\mathcal{U}$ becomes l-dimensional Hamming space, $\mathbb{H}^l = \{0,1\}^l$, consisting of all binary strings of fixed length l:

$$\mathbb{H}^l = \{\underbrace{000...000}_{l}, \underbrace{000...001}_{l}, \ldots, \underbrace{111...111}_{l}\}$$

Hence the size of this space is 2^l. The most popular matching rules used in this case are:

(a) r-contiguous bit rule [5], or
(b) r-chunks [1].

Both the rules say that a detector bonds a sample (i.e. data) only when both the strings contain the same substring of length r. To detect a sample in case (a), a window of length r ($1 \leq r \leq l$) is shifted through censored samples of length l. In case (b) the detector $t_{i,\mathtt{s}}$ is specified by a substring $\mathtt{s}$ of length r and its position i in the string. Below an example of matching a sample by r-detector (left) and r-chunk for affinity threshold $r = 3$ is given

$$\overbrace{1\,0\,\mathbf{0}\,\mathbf{0}\,\mathbf{1}\,1\,1\,0}^{l} \quad \leftarrow \text{sample} \rightarrow \quad \overbrace{1\,0\,\mathbf{0}\,\mathbf{0}\,\mathbf{1}\,1\,1\,0}^{l}$$
$$0\,1\,\underbrace{\mathbf{0}\,\mathbf{0}\,\mathbf{1}}_{r}\,0\,0\,1 \leftarrow r\text{-detector; } r\text{-chunk} \rightarrow *\,*\,\underbrace{\mathbf{0}\,\mathbf{0}\,\mathbf{1}}_{r}\,*\,*\,*$$

Here it was assumed that irrelevant positions in a string of length l representing the r-chunk $t_{3,\mathtt{001}}$ are filled in with the star ($*$) symbol. This way r-chunk can be identified with schemata used in genetic algorithms: its order equals r and its defining length is $r-1$. Although a single r-detector recognizes much more strings than a single r-chunk, this last type of detector allows more accurate coverage of the $\mathcal{N}$ space [1].

Further, the notion of the ball of recognition allows to define "optimal" repertoire $\mathfrak{D}$. Namely it consists of the detectors located in $\mathbb{H}^l$ in such a way that they cover the space $\mathcal{N}$ and their balls of recognition overlap minimally. A solution to such stated problem was given in [14]. To construct the r-detectors we split all the *self* strings into the templates represented identically as the r-chunks and we construct the detectors by gluing these r-chunks that do not belong to the set $\mathcal{S}$. More formally, if $t_{i,\mathtt{s}}$ and $t_{j,\mathtt{w}}$ are two candidate r-chunks, we can glue them if both the substrings are identical on $r-1$ positions starting from position $i+1$.

Using such an optimality criterion we come to the conclusion that shortest detectors are more desirable as they are able to detect more samples. However, Stibor [12] showed the coherence between r and l values for various cardinalities of S in terms of the probability of generating detectors, P_g. He distinguished three phases:

- Phase 1 (for lower r) – the probability P_g is near to 0,
- Phase 2 (for middle r) – the probability P_g rapidly grows from 0 to 1 (so called *Phase Transition Region*),

– Phase 3 (for higher r) – the probability is very near to 1.

Hence, we should be interested in generating detectors with medium length r (belonging to the second region) and eventually with larger values of r if the coverage of $\mathcal{N}$ is not sufficient.

In case of software approach, the detectors can not be too long, due to exponential increase in the duration of learning process, which should be finished in reasonable time. That was a main reason why b-detectors were not used to solve problems in high-dimensional datasets. In litetaure, we find that longest binary samples ($l = 49$) were used to model the system for monitoring TCP SYN packets to detect network traffic anomalies (called LISYS) [8]. In Section 3, we show, that detectors even with $l = 95$ can be generated in just a few ticks of clocks in case of hardware implementation of *NSA* algorithm.

3 Hardware implementation

Traditional software firewalls when analyzing and filtering packets flowing through the network use the processing power on which they are installed. This can lead to a significant reduction in responsiveness of the computer during a network attack. An alternative to software solutions are hardware firewalls, however, they are relatively expensive to use.

Another solution are firewalls embedded in a reprogrammable FPGA (Field Programmable Gate Array) architectures characterized by a high degree of flexibility during the design process. Usage of HDL (Hardware Description Language) languages reduces the cost of design and allows to transfer design between different architectures from manufacturers such as Xilinx or Altera. The use of reprogrammable architecture for the construction of a firewall reduces the cost of the final solution and increases the number of classified packets compared to pure software solution.

Construction of FPGA allows for parallelization of calculations and algorithms, so that they have wide range of applications during the HPC process (High Performance Computing). Another advantage of FPGAs is their low power consumption compared to the GPU (Graphics Processing Unit) solutions used in HPC, so that they represent better value performance-per-watt power consumption [13].

The use of FPGAs significantly shortens time to market (the length of time since the inception of the product concept to placing it on the market) compared to system based on ASIC (Application Specific Integrated Circuit).

The biggest advantage of FPGAs is that they can be reprogramed even after they have been installed in the target system. This allows to correct the errors or complement the design with new functionality. In the case of ASIC the process of design and manufacturing should be repeated and then replace a malfunctioning chip in the target system, which is associated with high costs.

The research are focused on building firewall with high throughput and latency as low as possible through the use of reprogrammable architectures. De-

signed in this way, the firewall will work independently without supervision of other (external) devices.

FPGAs contain programmable logic (configurable logic blocks) contains a set number of LUTs (LookUp-Table - small memory generate boolean function), flip-flops and multiplexers connected via programmable interconnects.

As the implementation language of hardware firewall has been chosen VHDL (Very High Speed Integrated Circuits Hardware Description Language) because is the most supported hardware description language by synthesis software and source code simulators.

3.1 Combinational component approach

In initial stage of work on the hardware firewall design was considered as a purely combinational circuit. The advantage of this approach is the clarity of its timing behavior (input/output response) analysis in oder to verify its correctness. Another advantage of this approach is the eases of making changes in the design description.

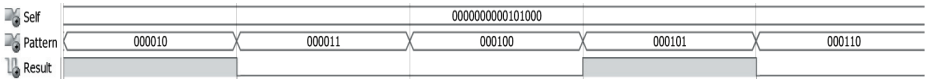

Fig. 1. Combinational component approach: simulation waveform (timing diagram) of anomaly detection.

Figure 1 shows a simulation of generating detectors for combinational component approach. In order to increase the readability of input/output values, results of operations component the were limited to a $l = 16$ for each sample and $r = 6$ for r-chunks. On simulation sample was market as Self - on the input might be inserted signal values of Self/NonSelf; detectors as $Pattern$ - r-chunks; and the $Result$ as the output of designed unit. $Value$ of $'1'$ on output denotes that the detector bonds a sample. For example, if $Self = \{0000000000101000\}$ and $Pattern = \{000010\}$ then $Result = 1$ ($Self$ and $Pattern$ are matched). The same behavior is observed for larger values $l = 64$ and $r = 32$ or $l = 128$ and $r = 64$ etc. The only restriction is length of $1 \leq r \leq l$.

Diagram on Figure 2 is schematic representation of the combinational unit design in terms of logic elements optimized to the target Xilinx Virtex 4 device xc4vfx12-12sf363. Technology schematic was generated for $l = 6$ and $r = 3$ (design occupies 10 LUT tables) because generating schema for lengths greater than the specified would lead to difficulties in the analysis. Detailed schematic generated for $l = 16$ and $r = 8$ occupies more than four pages and 53 LUT tables.

The proposed solution indicates very regular structure. Therefore, a different approach - pipelined - should be considered.

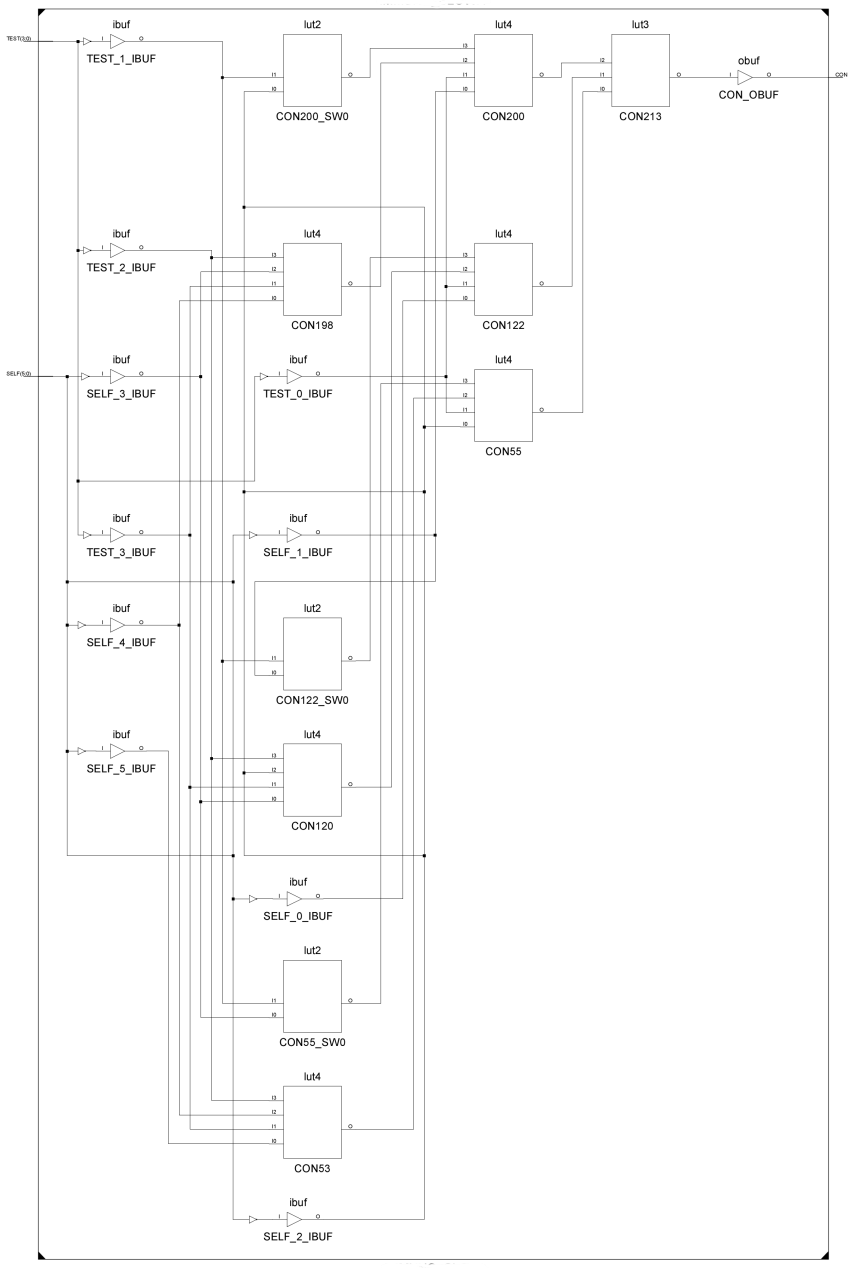

Fig. 2. Combinational component schema mapped into technology.

3.2 Pipelined approach

Throughput and delay are two critic performance criteria for designed component. Delay is the time that one task required to be completed, and throughput is the number of tasks that can be completed in one unit time. The major step of adding pipeline to immunologic anomaly detection design into stages. As was mentioned before combinational version indicates very regular structure. Comparison of r-chunk vector witch parts of self/nonself vector is realized on LUT tables which have the same or very close critical propagation delay time. An analogous situation occurs with the logical summation of the results of the comparison which is realized by a cascade connected LUT tables. Therefore when converting the combinational version of immunologic design to pipeline component version the system delay would not change but the system throughput will increase.

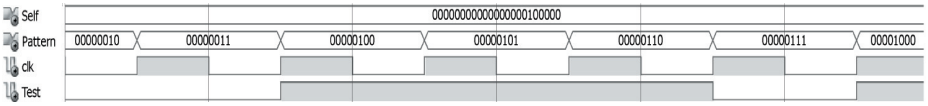

Fig. 3. Pipeline approach: simulation waveform (timing diagram) of anomaly detection.

Figure 3 shows a simulation censoring samples for pipeline component approach. In order to increase the readability of input/output values results of operations component the were limited to a $l = 23$ for each sample and $r = 8$ for r-chunks. Clk is a signal used for computing stages synchronization. Test signal value is delayed for 4 ticks of the clock. It means that the system needs 4 ticks of the clock to calculate value of Test signal and its directly depend on length of l and r.

3.3 Comparison of hardware implementations

Table 1. Comparison of combinational and pipeline versions.

l	r	tc $[ns]$	tp $[ns]$	n
11	8	9.101	7.234	3
23	8	11.133	6.869	4
71	8	20.201	7.486	5
35	32	11.801	7.454	4
47	32	18.335	7.215	5
95	32	14.615	7.906	6

Table 1 shows comparison of two approaches, combinational and pipeline, of matching patterns and samples. All measurements were made for a Xilinx

xc3s250e-5pq208 device from Spartan3E family. Parameter tc describe maximum combinational path delay for combinational version of matching system, tp - maximum combinational path delay for pipeline version and n how many ticks of the clock is needed to compute output value. It means that for the length $l = 47$ and $r = 32$ combinational system version may match 54M of sample/pattern pairs and pipeline - 138M - but the system response take 5 clock ticks (36,075ns).

4 Conclusions

Hardware implementations of negative selection algorithm are the way to overcome the scalability problem. The main advantage of this solution is a very short duration of both processes: generating detetors and censoring, as the most complex operations can be parallelized and aditionally computed without CPU utilization. Hence, in comparing to software approach, b-detectors can be generated very fast even for very long binary representations. Also, the result the overall duration of classification was significantly reduced and makes this solution possible to apply in on-line filtering the network connections.

There were compared two hwardware implementations: combinational and pipeline. Pipeline is commonly used technique to increase the performance of the design. Through a combination of pipelining and parallel processing, which is an advantage of FPGA devices, increases number of classified packets in network traffic using immunological methods.

This article is a first step for building dedicated (unique system security) hardware firewall with abbility to detect new types of attacks. In next step, implementation of V-Detector algorithm on FPGA will be performed to build hardware version of b-v model.

Acknowledgment

This work was supported by Bialystok University of Technology grants S/WI/3/13 and MB/WI/1/2014.

References

1. Balthrop J., Esponda F., Forrest S., Glickman M.: Coverage and generalization in an artificial immune system. In Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2002), New York, 9-13 July 2002, pp. 3–10.
2. Chu P. P.: RTL Hardware Design Using Vhdl: Coding For Efficiency, Portability, and Scalability. Wiley-Interscience, 2006.
3. de Castro, L., Timmis, J.: *Artificial Immune Systems: A New Computational Intelligence Approach*, Springer-Verlag, 2002
4. Chmielewski A., Wierzchoń S. T.: Hybrid negative selection approach for anomaly detection. In: A. Cortesi et al. (Eds.) Computer Information Systems and Industrial Management, LNCS 7564, Springer, 2012, pp. 242-253.

5. Forrest S., Hofmeyr S. A., Somayaji A., Longstaff T. A.: A sense of Self for Unix Processes. Proc. of the 1996 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, 1996, pp. 120–128.
6. Forrest S., Perelson A., Allen L., Cherukuri R.: Self-nonself discrimination in a computer. In Proc. of the IEEE Symposium on Research in Security and Privacy, Los Alamitos, 1994, pp. 202–212.
7. Harmer P. K., Wiliams P. D., Gunsch G. H., Lamont G. B.: Artificial immune system architecture for computer security applications. IEEE Trans. on Evolutionary Computation, Vol. 6, 2002, pp. 252–280.
8. Hofmeyr, S., Forrest, S.: Architecture for an Artificial Immune System. Evolutionary Computation J. vol. 8(4), 2000, 443-473.
9. Ji Z., Dasgupta D.: Real-valued negative selection algorithm with variable-sized detectors. Genetic and Evolutionary Computation GECCO-2004, Part I, LNCS Vol. 3102, Seattle, WA, USA, Springer-Verlag, 2004, pp. 287–298.
10. Ji Z., Dasgupta D.: *Revisiting negative selection algorithms*, Evolutionary Computation, vol. 15(2), 2007, 223-251.
11. Sayood, K.: *Introduction to Data Compression*. Elsevier, 2005
12. Stibor T.: Phase transition and the computational complexity of generating r-contiguous detectors. In Proc. of 6th International Conference on Artificial Immune Systems, LNCS 4628, 2007, pp. 142–155.
13. Vanderbauwhede W., Benkrid K.: High-Performance Computing Using FPGAs. 2013.
14. Wierzchoń S. T.: Generating optimal repertoire of antibody strings in an artificial immune system. In: M.A. Kłopotek, M. Michalewicz, S.T. Wierzchoń, eds: *Intelligent Information Systems. Proc. of the IIS'2000 Symposium, Bystra, Poland, June 12-16, 2000*. Springer 2000, pp. 119-133