Embedding the V-Detector Algorithm in FPGA

Maciej Brzozowski, Andrzej Chmielewski

Embedding the *V*-Detector algorithm in FPGA

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 *b*-*v* model is a hybrid immune-based approach for detecting anomalies in high-dimensional datasets. It is based on a negative selection algorithm and utilizes both types of detectors to achieve better results in comparison to single detection models. Also, it is an interesting alternative to well known traditional, statistical approaches, because only positive (*self*) examples are required at the learning stage. As a result, it is able to detect even unnkown or never met anomalies and this fact is one of the most attractive features of this approach. However, especially in the case of on-line classification, not only high accuracy but also high efficiency is needed. Thus, we propose to embed some complex tasks in a reprogrammable FPGA to offload CPU and speed up the classification process. This paper presents a hardware implementation of the *V*-Detector algorithm, which is the most complex and time consuming part of *b*-*v* model.

Keywords: artificial immune system, anomaly detection, FPGA

1 Introduction

An efficiency of *Natural Immune System* (NIS) is unsurpassed for all protection systems and verified over millions of years by living organisms. It is a very complex system focused on discrimination between own cells (called *self*) and pathogens (called *nonself*), which should be detected and eliminated. A nice feature of NIS is that it does not need any example of *nonself* samples to detect them as only the information about its own cells is sufficient. Hence, every organism has a unique "protection system", capable of detecting even a new type of attack and tolerates only own cells which form its body.

This dedicated and highly efficient protection system against various types of pathogens was an inspiration for developing *Artificial Immune Systems* (AIS). Within this domain, many types of algorithms were proposed, mainly focused on computer system security solutions. However, the most popular is *Negative Selection Algorithm* (*NSA*) [7] with the ability of detecting novel, never met samples, a counterpart of pathogens. Based on deep investigations with various types of large and high-dimensional datasets, a solution called the *b*-*v* model [6] was proposed. It minimizes the problem of scalability, by involving both types of receptors: *b*- and *v*-detectors. This hybrid approach, presented by conducting

numerous experiments, provides much better results in comparison to single detection models as well as traditional, statistical approaches, even though only positive (*self*) examples are required at the learning stage. It makes this approach an interesting alternative for well known classification algorithms, like SVM, k-nearest neighbours, etc. The b-v model is briefly described in Subsection 2.3.

However, there are lots of domains where not only high accuracy is crucial. In some cases, very high efficiency is also required to examine all samples in a short time. On-line classification systems like firewalls, intruder detection and prevention systems, etc. are examples of applications which usually operate on huge amounts of data and over-lengthy delays, related with data processing, are unacceptable.

Thus, every way to increase the speed of detection process is highly desirable. One of them is embedding algorithms in a reprogrammable FPGA (Field Programmable Gate Array). Our preliminary results [3] conducted using b-detectors, have shown while generating them as well as during censoring samples, which are the most complex operations, can be parallelized and additionally computed without CPU utilization. As a result, censoring of millions samples took only a few ticks of the clock (a few nanoseconds).

In this paper we present our approach to embedding the V-Detector algorithm in FPGA as being a very important part of b-v model. All samples not recognized by very fast b-detectors are passed to V-Detector which is responsible for the final decision: censored sample is *self* or *nonself*. It operates on real-valued detectors (called v-detectors), in contrast to fast b-detectors represented as a binary string. Hence, this step is the most complex and time consuming in the whole process of detecting anomalies, especially in the case of high-dimensional datasets. Hardware implementation of this algorithm should significantly increase its efficiency.

2 Negative Selection

The NSA, proposed by Forrest *et al.*, [8], 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 the collection of all *self* molecules and its complement $\mathcal{N}$ in $\mathcal{U}$ represents all *nonself* molecules. Let $\mathfrak{D} \subset \mathcal{U}$ stand 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 modelled 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 e.g. in [9] and [11].

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 the 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 the sufficient number of detectors will be generated.

A perfect *NSA* should cover whole $\mathcal{N}$ space by sets of detectors to detect all *nonself* samples. However, as shown by numerous experiments, scalability is a key issue here, regardless of the representation of samples used [12][6]. This problem is also discussed in this article.

2.1 Detectors and algorithms

Generally, there are two type of receptors used:

- b-detectors - represented as binary string (Hamming space),
- v-detectors - represented as real-valued vectors.

For each of them, many algorithms with different matching rules were proposed. For binary representation, the most popular were presented in [7] and [2]. In the case of real-valued vectors, V-Detector algorithm [10] is the most known. Hardware implementation of this algorithm, described in details in Subsection 2.2, is the main subject of this article.

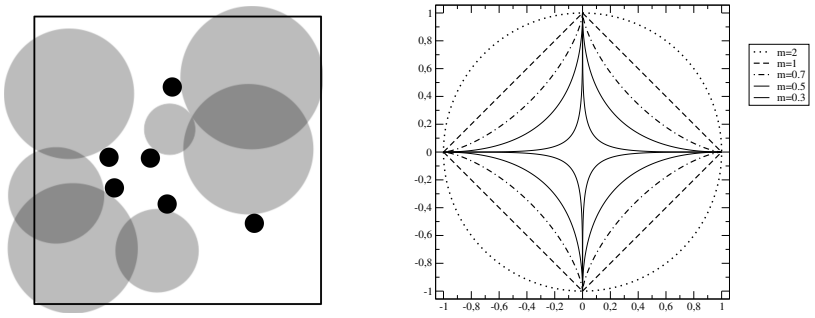
Because, neither b- nor v-detectors were not capable to detect anomalies to a satisfactory degree, a b-v model here was proposed [6] (see Subsection 2.3), which used both types of detectors to overcome the problem with scalability. As shown in the performed experiments, it provides much better results in comparison to single detection models as well as in comparison to traditional, statistical approaches, even though only positive self examples are required at the learning stage.

2.2 Real-valued representation

To overcome scaling problems inherent in Hamming space, Ji and Dasgupta proposed a real-valued *NSA*, termed *V-Detector* [10].

It operates on normalized vectors of real-valued attributes; each vector can be viewed as a point in the d-dimensional unit hypercube, $\mathcal{U} = [0, 1]^d$. Each *self* sample, $s_i \in \mathcal{S}$, is represented as a hypersphere $s_i = (c_i, r_s)$, $i = 1, \ldots, l$, where l is the number of *self* samples, $c_i \in \mathcal{U}$ is the center of s_i and r_s is its radius. It is assumed that r_s is identical for all s_i's. Each point $u \in \mathcal{U}$ inside any *self* hypersphere s_i is considered as a *self* element.

The detectors d_j are represented as hyperspheres also: $d_j = (c_j, r_j)$, $j = 1, \ldots, p$ where p is the number of detectors. In contrast to *self* elements, the radius r_j is not fixed but it is computed as the Euclidean distance from a randomly



(a) (b)

Fig. 1: (a) Example of performance V-Detector algorithm for 2-dimensional problem. Black and grey circles denotes *self* samples and v-detectors, respectively. (b) Unit spheres for selected L_m norms in 2D.

chosen center c_j to the nearest *self* element (this distance must be greater than r_s, otherwise the detector is not created). Formally, we define r_j as

$$r_j = \min_{1 \leq i \leq l} dist(c_j, c_i) - r_s. \tag{2}$$

The algorithm terminates if a predefined number p_{max} of detectors is generated or the space $\mathcal{U} \backslash \mathcal{S}$ is sufficiently well covered by these detectors; the degree of coverage is measured by the parameter co – see [10] for the algorithm and its parameters description.

In its original version, the V-Detector algorithm employs Euclidean distance to measure proximity between a pair of samples. Therefore, *self* samples and the detectors are hyperspheres (see Figure 1(a)). Formally, Euclidean distance is a special case of Minkowski norm L_m, where $m \geq 1$, which is defined as:

$$L_m(\mathrm{x}, \mathrm{y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^m \right)^{\frac{1}{m}}, \tag{3}$$

where $\mathrm{x} = (x_1, x_2, \ldots, x_d)$ and $\mathrm{y} = (y_1, y_2, \ldots, y_d)$ are points in $\Re^d$.
Particularly, L_2-norm is Euclidean distance, L_1-norm is Manhattan distance, and L_∞ is Tchebyshev distance.

However, Aggarwal *et al.* [1] observed that L_m-norm loses its discrimination abilities when the dimension d and the values of m increase. Thus, for example, Euclidean distance has the best (among L_m-norms) metrics when $d \leq 5$. For higher dimensions, the metrics with lower m (i.e. Manhattan distance) should be used.

Based on this observation, Aggarwal introduced *fractional distance metrics* with $0 < m < 1$, arguing that such a choice is more appropriate for high-dimensional spaces. Experiments, reported in [5], partially confirmed the effi-

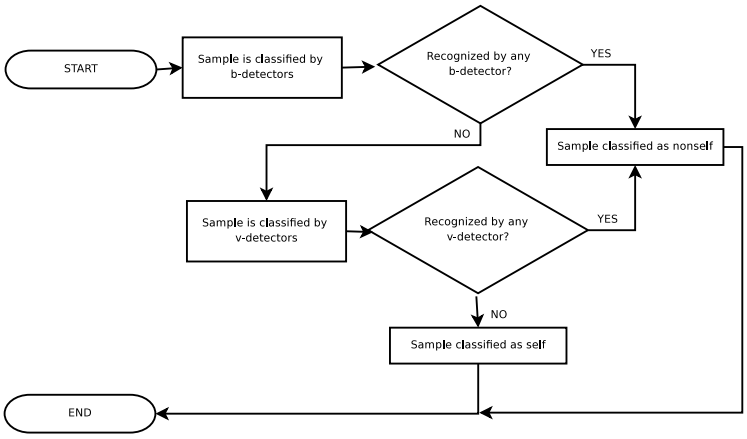


Fig. 2: Flow diagram of the classification process for b-v model.

ciency of this proposition. For $0.5 < m < 1$, more samples were detected, in comparison to L_1 and L_2 norms. However, for $m < 0.5$ the efficiency rapidly decreased and for $m = 0.2$, none samples were detected. Moreover, these experiments also confirmed a trade-off between efficiency, time complexity and m. For fractional norms, the algorithm runs slower for lower m values; for $L_{0.5}$ the learning phase was even 2-3 times longer than for L_2.

Another consequence of applying fractional metrics for V-Detector algorithm is modification of the shape of detectors. Figure 1(b) presents the unit spheres for selected L_m-norms in 2D with $m = 2$ (outer most), $1, 0.7, 0.5, 0.3$ (inner most).

2.3 Brief description of the b-v model

The b-v model [6] is the only *NSA*, involving both types of detectors, namely b- and v-detectors. It was designed for anomaly detection in high-dimensional data, which are difficult to analyze due to the lack of appropriate similarity metrics which enable the covering of space $\mathcal{N}$ in sufficient degree and reasonable time. One of the important features of the b-v model is its ability to minimize the overlapping regions between sets of detectors. We do not expect that b-detectors cover the space $\mathcal{N}$ in sufficient degree, as it can consume too much time. On the other hand, fewer numbers of v-detectors should be generated because some part of space $\mathcal{N}$ is already covered by b-detectors. The main idea of this algorithm is depicted in Figure 2.

Here, the b-detectors, as those providing fast detection, are used for preliminary filtering of samples. The samples which did not activate any of the b-detectors are censored by v-detectors next. As a result the overall duration of classification could be significantly reduced as fewer v-detectors were needed to cover space $\mathcal{N}$. Hence, this model is more efficient for on-line classification systems in comparison to standard negative selection approaches which are based only on one type of detectors.

Embedding the V-Detector algorithm in FPGA (see Section 3) should significantly affect the efficiency of the entire b-v model. This also opens the possibility of applying them in other domains, including devices used in the Internet of Things technology.

3 Hardware approach to V-Detector algorithm

As was already mentioned, classification with the V-Detector algorithm is a very complex and time consuming process because of its high complexity of performed calculations that need to be done on real-valued vectors. Time of classification of a single sample directly depends on its dimensionality ($\Re^d$) and number of detectors, because for each single sample, there is a need to calculate distance to generated detectors from set $\mathfrak{D}$. The described process, of course, might be divided into sub tasks to parallelize classification. As a result, the duration of this process should be significantly decreased. However, when calculations are performed on CPUs or GPUs processors, its efficiency is strictly restricted by number of processors and numbers of its cores.

One of the solutions, corresponding to the above restrictions, are programmable devices, especially Field Programmable Gate Array (FPGA) devices, providing a high performance, low power consumption and relatively low price in comparison to CPU/FPU solutions. FPGA is mainly designed to parallelize computational tasks. Therefore, it might be used in projects where the main indicator is performance. Designers equipped FPGA devices in specialized IP (Intellectual Property) blocks like multipliers, pre-adders and accumulators for increasing the number of computations per second and other more complicated blocks like embedded CPUs, DSP, Ethernet Physical Interfaces, PCI Express, DRAM controllers and many others. Moreover, some FPGA devices allow for the partial reconfiguration (reprogramming) during its operation - helping the system to adopt to rapidly a changing environment.

FPGA devices are characterized by high computing power, flexibility and scalability. They can be adapted as a base for on-line classification systems eg. firewalls and intruder detection systems for home use as well as for enterprise solutions.

As was presented in [3], b-detectors were successfully implemented in reprogrammable architectures, especially in FPGA devices. Hence, it is natural to try to do the same for the V-Detector algorithm, where we could expect more spectacular results in comparison to a software approach, which was a bottleneck in the b-v model.

Classification algorithm will be explained in further sections but firstly we must present how the distance between detectors and samples is calculated. The main difference, in comparison to software implementation, is the usage of hypercubes, instead of hyperspheres. Such a shape is more suitable for reprogrammable architecture.

Each *self* sample, $s_i \in \mathcal{S}$, is represented as a hypercube $s_i = (c_i, l_s)$, $i = 1, \ldots, l$, where l is the number of *self* samples, $c_i \in \mathcal{U}$ is the center of s_i and l_s

is half of the length of its side (edge). It is assumed, that l_s is identical for all *self* samples. Similar to software approach, each hypercube $u \in \mathcal{U}$ inside any *self* hypercube s_i is considered as a *self* element.

The detectors d_j are represented as hypercubes also: $d_j = (c_j, l_j), j = 1, \ldots, p$ where p is the number of detectors. In contrast to *self* elements, the half length of its side l_j is not fixed, but it is computed as the distance from a randomly chosen center c_j to the nearest *self* element (this distance must be greater than l_s, otherwise detector is not created).

Formally, we define l_j as:

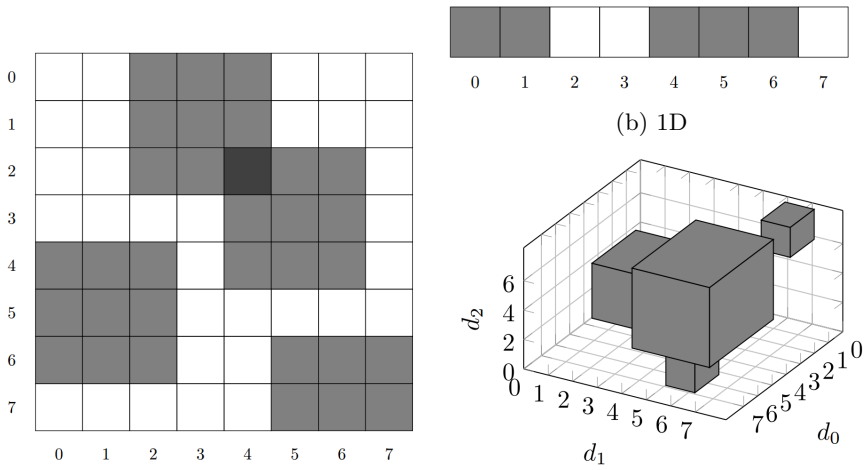
$$l_j = \min_{1 \leq i \leq l} dist(c_j, c_i) - l_s, \tag{4}$$

where distance between two centers of hypercubes x and y is defined as:

$$dist(x, y) = \min_{1 \leq i \leq d} |x_i - y_i|. \tag{5}$$

Each hypercube is axis-aligned. It means, rotation of hypercube is not allowed. Two hypercubes are not overlapping when:

$$\min_{1 \leq i \leq d} |x_i - y_i| > l_x + l_y. \tag{6}$$



(a) 2D (b) 1D (c) 3D

Fig. 3: Example of axis-aligned v-detector hypercube in $\Re^d$.

Figure 3 presents d-dimensional hypercubes. In Figure 3(b) two detectors $d_1 = ([0], 1)$, $d_2 = ([5], 1)$ in one dimensional space are marked (grey colour). In Figure 3(a) two detectors $d_1 = ([1, 3], 1)$ and $d_2 = ([3, 5], 1)$ are overlapping.

In Figure 3(c): $d_1 = ([1,1,1],0)$, $d_2 = ([0,6,7],0)$, $d_3 = ([5,5,0],0)$ and $d_4 = ([5,5,5],1)$ are examples in three dimensional space. In V-Detector classification process it is sufficient to state that sample overlap with one of the detectors. There is no need to state that detector includes the sample (sample overlaps a whole over the detector or overlaps its partially). In both cases, the sample should be rejected.

3.1 Emebedded *V*-Detector - the fastest approach

The first of the proposed solutions, denoted as V-Detector$-HFast$, for reprogrammable architectures holds the entire $\mathfrak{D}$ in target device resources. As a result, it should maximize the speed of classification process for V-Detector algorithm. The base concept allows the determination of distance between the single sample and the entire set of $\mathfrak{D}$ in one cycle of the designed system. Moreover, the classification process does not depend to such an extent on the number of dimensions as in the classical approach. Here, the universe set $\mathcal{U}$ is represented by bitmap with detectors modeled as hypercubes. The number of distinguish samples (represented as vectors) depends on the size of used internal memory.

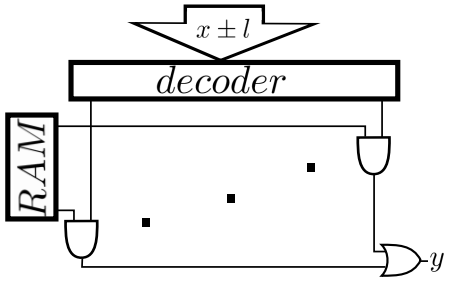


Fig. 4: V-Detector - the fastest approach for reprogrammable architectures.

Memory and other arithmetic operations are mapped into logic elements available on the reprogrammable device, in our case FPGA. For each dimension, a decoding vector responsible for choosing data from set $\mathfrak{D}$ is created, depending on the sample's coordinates. For each coordinate logical products are created of decoding vectors (for all dimensions) and its corresponding memory values. In the last step, the number of '1' in the logical product is counted. If this value is not equal to 0, it means, sample overlaps at least one detector. Figure 4 shows a simplified diagram of V-Detector (for one dimension) based on internal FPGA memory.

The proposed solutions for reprogrammable architectures is incredibly fast in classification of both single and group of samples. This approach, apart from the high-speed classification, also has some disadvantages. Set $\mathfrak{D}$ is held in FPGA internal RAM (very limited capacity) or is mapped in the logical area on the

device. In this case, a synthesized classification algorithm is highly demanded for device logic resources. Therefore, we are limited by logic element numbers on the target programmable device, in which one, design has to be mapped. In this case we have to choose a target device with appropriate logic elements overhead. The lack of adequate number of logical resources may lead to a decrease in the number of analyzed dimensions, or enforces the reduction of the resolution of the analyzed samples.

3.2 Embedded V-Detector - approach based on external memory

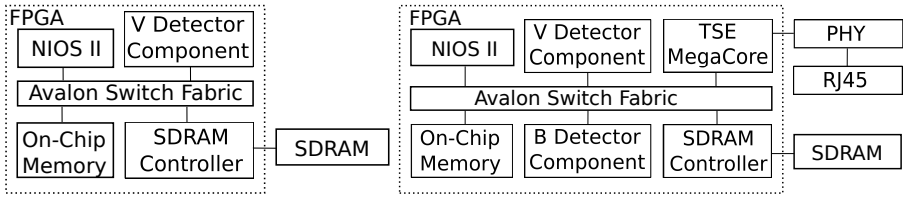
The presented classification solution, denoted as V-Detector$-HRAM$, is based only on the resources provided by the target device, is characterized by constraints on the system resolution. One of the methods to increase the resolution of the system is to move the set of detectors from the internal device resources to external memory. In this case, part of the available resources has to be destined to calculate the read/write memory cell addresses. The designed component informs which memory lines are needed in the calcification process of each sample and chooses only the appropriate cells from them. The proposed solution is slowed down by time needed by external memory to perform write and read operations compared to the approach presented in Subsection 3.1. When the system is properly scaled to expected samples (in numbers of dimensions and numbers of bits per dimension) i.e. $l_s = 0$ (is small as possible) classification process should need only one read operation from external memory.

For DE2-115 Development Board we were able to achieve a resolution of 2^{30} distinguishable samples.

4 Hardware V-Detector approaches evaluation

In the proposed solutions, the learning phase is simplified, in comparison to software implementation. The optimization process of $\mathfrak{D}$ set is not needed because its size does not affect the duration of classification of a single sample. Bitmap hypercube representation of $\mathcal{U}$ is constant and therefore the number of detectors has no impact on its size. Classification time is shorter because the designed components consider calculations only in the nearest surroundings of the sample. All the necessary calculations are performed at the level of the FPGA. Therefore, the designed component may process large amounts of data in a single period of time.

Both presented solutions were implemented and synthesized for Altera Cyclone IVE EP4CE115F29C7 device equipped on the DE2-115 Development and Education Board by using Quartus II 15.0. Figure 5(a) shows simplified diagrams of the actual designs. The core of the solution is Nios II processor with connected components via an Avalon bus. In the first stage, components were tested as operated independently. In the second stage, components were integrated into the system (Figure 5(a)). In the future, both systems, V-Detector



(a) (b)

Fig. 5: System diagrams for FPGA device.

Table 1: Comparison of average duration of classification process for V-Detector$-HFast$ and V-Detector$-HRAM$ algorithms for different number of bits per dimension (n) for $Iris$ dataset.

	$HFast$		$HRAM$			
	$n=2$		$n=2$		$n=6$	
	time[ms]	time (clock)	time[ms]	time (clock)	time[ms]	time (clock)
Iris-setosa	0.18	17914	0.58	58181	0.58	58181
Iris-versicolor	0.18	17914	0.58	58061	0.58	58061
Iris-virginica	0.18	17914	0.58	58038	0.58	58038

based on internal and on external RAM cells, will be equipped with two Ethernet interfaces 10/100/1000 and tested in a network environment (Figure 5(b)). Another possibility of the further development is integrated design with PCI Express and use also of the desktop computer resources.

To measure performance (profiling) of the designed systems we used Altera Megacore Performance Counter Unit. The designed system was based on Nios II worked with 100MHz frequency.

As was already mentioned in Subsection 3.1, the dimensionality of samples which can be processed by V-Detector$-HFast$ algorithm is highly restricted by resources availability in target devices. In the case of Altera Cyclone IVE EP4CE115F29C7, which was used in our experiments, internal memory capacity is limited to 3888kb. Such a restriction forced us to select a rather small dataset for testing. We chose the most popular dataset used for classification purposes, namely $Iris$, which consists of only 150 samples. Each of 4 attributes was represented by integer value from the range 0-3 (2 bits). Detectors were generated, assuming that one class of thew iris plant is regarded as set $\mathcal{S}$ during the learning stage. In all cases, samples were successfully classified. The main advantage of the hardware approaches was the duration of classification, and was presented in Table 1.

We can compare those values with the average time of classification achieved for software implementation executed on a PC equipped with Intel i7-3770 processor (8 cores) 3.4GHz with 16GB RAM. In this case, process censoring all samples took about 3 ms. It is about 20 times slower than V-Detector$-HFast$, and about 6 times slower than V-Detector$-HRAM$. Taking into consideration

Table 2: Comparison of average duration of classification process for software implementation and V-Detector$-HRAM$ algorithm applied to randomly selected subsets for KDD Cup 1999 dataset.

$Software$	$HRAM$	
time[ms]	time[ms]	time (clock)
106 000	45	4483698

that the frequency of the CPU on the PC computer was 340 times higher, it can be easily computed how fast the hardware approach is.

Table 1 contains also the results of other classification experiments with the same dataset. Here, samples were represented as 4 dimension vectors with 6 bits per dimension. For V-Detector$-HFast$ this design was too big to map into available architecture. Thus experiments were conducted only for the V-Detector$-HRAM$ algorithm for which the classification process took 0.58 ms (the same time as in first experiment). Here, classification strictly depends on access time to external memory (read/write data operation). It is clear that all operations performed on internal resources are faster than on external memory. The increase in the number of dimensions does not significantly affect the time of classification. The most important is a properly scaled system (sample size).

However, the most spectacular results were obtained for some subsets of KDD Cup 1999 from UCI Machine Learning Repository. This database was too big for the current implementation of V-Detector$-HRAM$. Hence, only the samples of the ICMP protocol were used for tests with randomly selected 7 of 41 attributes (each coded on 4 bits). In this way, our test set contains 1074994 unique samples and an average of 683 detectors were used. During the tests we could observe, the duration of classification was more than 2000 times faster in the case of hardware implementation (see Table 2).

5 Conclusions

In this paper we present our approach to hardware implementation of the V-Detector algorithm, which is the most computationally complex part of b-v model. Our preliminary promising results with b-detectors was a first step for building a very fast classification system embedded in reprogrammable devices. To achieve this goal, we had to implement also the V-Detector algorithm, which operates on computationally complex real-valued vectors.

Here, we presented two different possible approaches, called V-Detector$-HFast$ and V-Detector$-HRAM$. Both of them have some disadvantages, which are mainly related with the limitation of used memory. However, V-Detector$-HRAM$ can be extended to be used with external memory, which can make this approach possible to be applied for much bigger datasets than those used in the experiments described in this paper.

The performed experiments confirmed that hardware implementations can significantly speed up the classification process, which usually is the most crucial

in classification systems. Obtained preliminary results are a good starting point to build a hardware version of the b-v-model, dedicated for an on-line immune-based intrusion detection system.

Acknowledgment

This research was partially supported by the grants S/WI/3/13 and MB/WI/1/2014 of the Polish Ministry of Science and Higher Education.

References

1. Aggarwal C. C., Hinneburgand A., Keim D. A.: On the surprising behavior of distance metrics in high dimensional space. LNCS, Vol. 1973, Springer, 2001, pp. 420–434.
2. 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.
3. Brzozowski M., Chmielewski A.: Hardware Approach for Generating b-detectors by Immune-Based Algorithms, Computer Information Systems and Industrial Management, 2014, LNCS 8838, Springer, pp. 615–623.
4. Chu P. P.: RTL Hardware Design Using Vhdl: Coding For Efficiency, Portability, and Scalability. Wiley-Interscience, 2006.
5. Chmielewski A., Wierzchoń S. T.: On the distance norms for multidimensional dataset in the case of real-valued negative selection application. Zeszyty Naukowe Politechniki Białostockiej, No. 2, 2007, pp. 39–50.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. Ji Z., Dasgupta D.: *Revisiting negative selection algorithms*, Evolutionary Computation, vol. 15(2), 2007, 223-251.
12. Stibor T., Mohr P.H., Timmis J., Eckert C.: Is negative selection appropriate for anomaly detection? GECCO 2005, pp. 321-328.
13. Vanderbauwhede W., Benkrid K.: High-Performance Computing Using FPGAs. 2013.