Immuno-repairing of FPGA designs

Norma Montealegre, Franz J. Rammig

Abstract FPGAs can be used for the design of autonomic reliable systems. Advantages are reconfiguration and flexibility in the design. However commercial FPGAs are first prone to errors. Second, the design flow is not yet supported for the use of fault tolerance techniques like Built-In Self-Tests. Fault tolerance can be reached through error detection and fault recovery. Most error detection techniques are not suitable for on-line detection because of detection times and long and inflexible training. This paper proposes a fault tolerant design for FPGAs. It has a Built-In Self-Test which error evaluation and fault recovery is supported by computing techniques inspired in the Immune System. A fault recovery and a hardware implementation model are also to be presented.

Keywords: autonomic systems, fault tolerance, immunocomputing, FPGA, BIST.

1 Introduction

Nowadays there is the demanding requirement of having systems which faults can be recovered without human intervention. That is the field of autonomic reliable systems. Autonomous robots and vehicles in outer space and undersea systems [8] are prone to errors due to its dynamic and environment of action. These systems are designed with radiation-hardened or higher and lower temperature range components, like radiation-hardened FPGAs [10]. Hardware design techniques based on Triple Modular Redundancy help in developing FPGA-based circuits resilient to SEUs (Single Event Upset) [7], like the tool referred in [11] or the TMR-Tool from Xilinx [21]. While circuits are hardened with special components and TMR has a limited fault recovery, a seamless design flow for fault recovery is not present yet.

Some algorithms were developed in the field of Artificial Immune Systems, inspired in the vertebrate's immune system. They have served in solving computing problems. Nevertheless those algorithms have inspired also electronic designs in the

Norma Montealegre
Heinz Nixdorf Institute, Fuerstenalle 11, 33102 Paderborn, Germany
e-mail: norma@upb.de

Please use the following format when citing this chapter:

Montealegre, N. and Rammig, F.J., 2008, in IFIP International Federation for Information Processing, Volume 268; *Biologically-Inspired Collaborative Computing*; Mike Hinchey, Anastasia Pagnoni, Franz J. Rammig, Hartmut Schmeck; (Boston: Springer), pp. 137–149.

searching of fault tolerance. One example is the research coined as ”Immunotronics” (Immune + Electronics), the hardware fault tolerance inspired by the immune system [3]. In [18] a centralized immune layer is presented. The learning phase identifies correct operation of the systems as ”self” building antibody patterns composed of: inputs, excitation, correct states and outputs. In the operational phase, the identification of ”non self” operations is implemented by means of genetic algorithms. On the other side, a decentralized immune layer has taken inspiration from cell biology to create a multicellular FPGA. This idea emerged the field of Embryonics which together with Immunotronics generate a two level structure [2]. The first level is composed of the embryonic cells which communicate across data channels. A second layer is composed by antibodies which communicate also across data channels and are named together the lymphatic network. Trans-layer communication channels are present between antibodies and embryonic cells as well. Antibodies store self-tolerance conditions. Furthermore, healing of the embryonic cells is regarded by [17], who considers a cell's self-test for the fault diagnosis, and a cell repair or elimination through reconfiguration of the cell's routing. All these methods carry to a new hardware conception not available in commercial FPGAs.

Immunocomputing explores, in a formal way, the principles of information processing that proteins and immune networks utilize in order to solve specific complex problems [16]. Free binding among proteins inspired Formal Immune Networks, which are able to learn, recognize and solve problems. This method is based in the Singular Value Decomposition of a matrix. It proved to have small learning and recognition times and a good resource efficiency regarding memory and computing [13]. Moreover, it presents a self-organizing property since, for a training set, iterations within the algorithm self-converge to antibodies [13]. Making use of its efficiency, Immunocomputing can be used for on-line error detection [15].

A self-test system has two main components: a test pattern generator and a test response evaluator. The test patterns and expected responses are stored in a memory. It is necessary a control signal to turn on the testing, a counter to address the memory and a comparator for comparing the obtained response with the expected one [9]. Because of the quantity of test patterns, this approach is time consuming. There are some alternatives of output response analysis in which output data compaction takes place. One of them are concentrators, counting techniques, signature analysis, accumulators, comparators, etc [12]. Comparison-based response evaluators compare on a vector-by-vector basis the expected responses stored in a memory and the output responses of the circuit under test. This approach is simple and modular. Besides a distributed Built-In Self-Test with n-test pattern generators, n-circuits under test and one test response evaluator can be applied. A BIST system can also work on-line [1]. The potential problem is the long time that may be required to cycle through the test patterns and evaluate the responses before determining if an error is present or not, [5] approaches this problem. This is critical for systems where the BIST works on-line and fault recovery should be done at time. A molecular approach is given by [4] and [19], but a circuit oriented design is not taken into account. Therefore a correct partitioning of the circuit, a distributed BIST with a fast response evaluator and fault recovery support is needed. This paper is a contribution following this tendency.

Fault tolerance can be reached through error detection and fault recovery. The present paper proposes a distributed fault tolerant design for FPGAs using Formal Immune Networks and self-test systems. The system has a distributed error detection mechanism through distributed Built-In Self-Tests inside a FPGA, see Fig. 1. BIST synthesis for a very large design may be possible within linear time by extracting sub-circuits which are almost constant in size [6]. That accelerate logic BIST synthesis procedures and reduces the time error detection takes. The circuit under test is one part of a partitioned circuit. The circuit receives a test pattern and the response is evaluated by means of cFINs (cytokine Formal Immune Networks) [14]. BIST can profit of the celerity offered by the cFIN method in detecting errors applying a determined error correction method at the proper time. Test response evaluation and fault recovery by cFIN for fault tolerant FPGA circuit designs is the main contribution of the present work. The decentralized BIST procedure is controlled by a global test scheduler module, a fault processing mechanism and a fault recovery module. A hardware implementation of the whole system is also proposed.

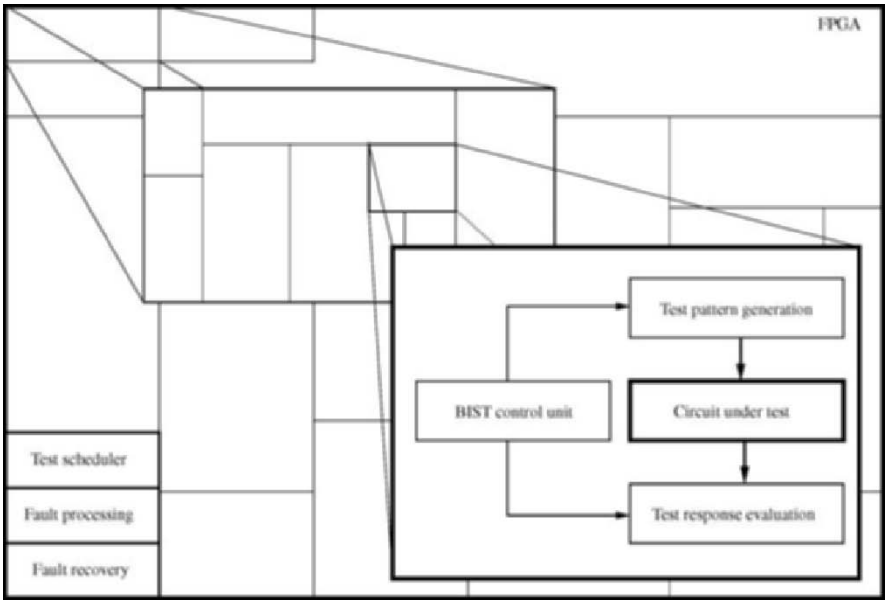

Figure 1 *Diagram of the proposed system*

2 Built-In Self-Test Proposal

Figure 2 shows a BIST proposal with on-line learning. The system needs to be trained with the test patterns before being operational. For test response evaluation purposes, outputs of the Circuit Under Test are evaluated with a method inspired in the cytokine-Formal Immune Networks and presented in Section 5. Making a biological analogy, an *antibody* represents the *expected output* transformed into the Formal Immune Network space. An *antigen* is the *response of the circuit under test*. A *cytokine* represents the *action* to be taken for fault recovery purposes.

It is important to note that after training of the system, on-line learning can take place. This is possible mapping the value of the new A_i test pattern into the cFIN space. This point is added to the compacted expected response data (compaction or compression performed by means of cFIN). Therefore, in case of a change in the training patterns, the training phase does not necessarily have to be repeated with the entire training set [13].

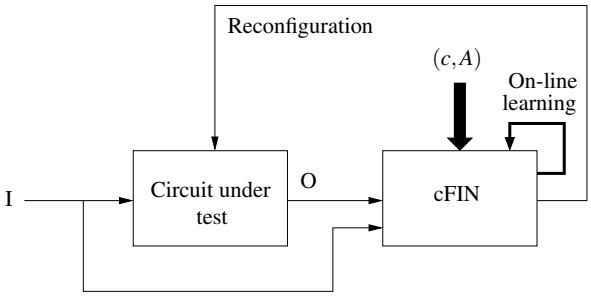

Figure 2 *Built-in-self-test with a cFIN on-line learning*

If this BIST model is applied for a whole circuit, the complexity in building the training matrix and the time for training and recognition may explode. Therefore, circuit partitioning [6] is considered, see Fig.1. Methods for circuit partitioning are not the scope of this paper.

A training matrix $V(c,A)$ should be provided prior to the operation of the system (test pattern generation). A is a matrix with information over expected responses under defined inputs. Each expected response should be linked to a recovery procedure in case of failure, expressed by c. In case of combinational circuits, training patterns are composed of Input/Outputs. But, sequential circuits consider also stimuli and internal states, as seen in Fig. 3. For test purposes, such circuits may be transformed to a sequence of combinational ones using conventional scan-path techniques.

The training matrix should regard the procedure for fault recovery under failure. For every training pattern is recommended to have a recovery alternative expressed in an integer coded value c, see Fig. 4. c represents a cytokine that signals the action to be taken at the time of finding an error.

The BIST Control Unit supports the hierarchical BIST strategy shown in Fig. 1. It contains an input for starting the BIST and an output for indicating the end of the test. A pattern counter determines the ending of a test. Two schemes to be considered are possible at the time of designing the BIST, the *test-per-scan* and the *test-per-clock*

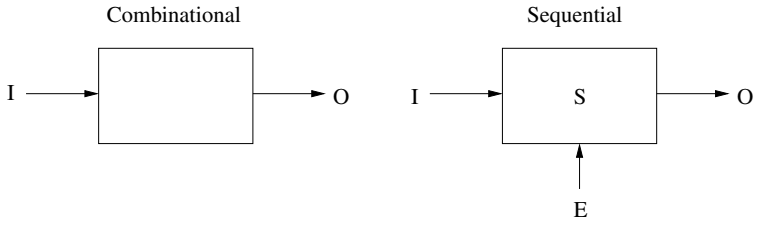

Figure 3 *Consideration at the moment of building the training matrix*

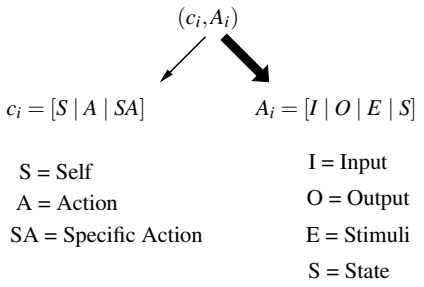

Figure 4 *Test cases*

scheme. It is not the aim of this paper to give a detailed description. Please refer to [20].

3 Global BIST

In order to apply a consistent error correction, the test schedule, fault processing and fault recovery are global modules for the whole system.

Depending on the application, tests can be recurrent or preemptive. In the case of a preemptive one, time error recognition should be considered in order to plan the frequency of testing. Frequency of testing is a function of the clock rate as well. A *test session* is a set of test unit processed in parallel and a *BIST schedule* is a series of test sessions which is implemented by the BIST Control Unit in hardware.

The class or cytokine's natural value represents the action to be taken in the design when a failure occurs. It has to be specified which recovery method should be applied for a specific failure. This data should be provided together with the training matrix in the case of *Supervised Learning* but it could also be determined after the training process by clustering points in the mapped FIN space. That is the case of *Unsupervised Learning*. In the first case this array can be constructed with the following data:

$$c = \{S \mid A \mid SA\} \tag{1}$$

Where:

S *Self* considers whether the recognized pattern is a failure "non-self" or a particular pattern. This can be used not only for failure detection, but also for warning states not considered as malicious.

A *Action* considers a general action to be taken i.e. total reconfiguration.

SA *Specific Action* considers a more refined method to recover from the failure.

Fault recovery is based on the reconfigurabilty property of FPGAs. Therefore, the failure recovery can be executed by total or partial reconfiguration. Other alternatives for fault recovery are application dependent and should be addressed at the time building the EA array.

4 Hardware Implementation

The proposed BIST can be implemented as an Intellectual Property Core inserted into the same FPGA as the circuit to be tested, Fig. 5. In this case, faults present in the Circuit Under Test are also prone to appear in the BIST. An alternative is to provide an external second FPGA which implements only this procedure.

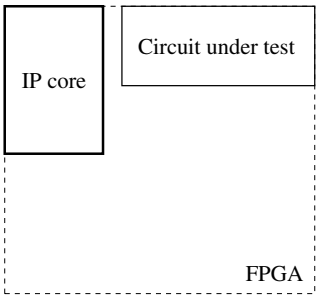

Figure 5 *Implementations as an IP core*

It is also possible to consider an external circuit composed of a DSP and a micro-controller, like the one in Fig. 6. The DSP is able to compute in parallel the mapping of points to the FIN and to compare distances among points [13]. The micro-controller could implement modules of the global BIST. Nevertheless, faults in the connection path between the circuit under test and the self-test system should be regarded in this implementation case.

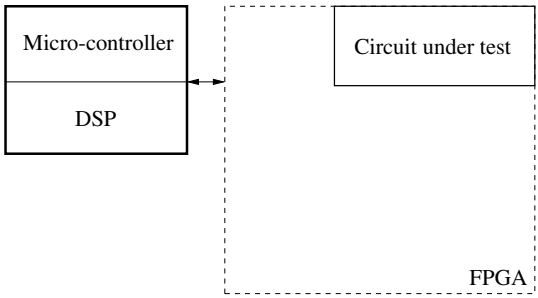

Figure 6 *Implementation as an external circuit*

5 c-Formal Immune Networks

This section explains the method of training and recognition of a cFIN. This method is used in the implementation of the response evaluation and fault processing of the Built-In Self-Testing system. For a more detailed and extensive explanation of this theory, please refer to [14], [13] and [15].

Immunocomputing intends to establish a new kind of computing. The main difference with other kinds of computing lays in its basic element, the *formal protein*. A protein is an essential component of organisms and participate in every process within cells. Proteins constitute epitopes present in antigens and antigen presenting cells. Proteins constitute also paratopes present in antibodies. Epitope is the minimum molecular structure that is able to be recognized by the immune system. One epitope matches with a paratope in molecular recognition. Figure 7 shows the antigen binding site of an antibody named as *paratope* that recognizes the epitope of an antigen or an antigen presenting cell. An antigen presenting cell is a cell that has digested an antigen and presents in its surface an epitope. An epitope is made of around 10 amino-acids. The same applies to a paratope. A protein is composed of amino-acids arranged in a linear chain. The 3D shape or tertiary structure of the epitope is recognized by a paratope, see Fig. 7. It means, an epitope is a kind of surface protein. That is why proteins will be seen as the basic element in Immunocomputing.

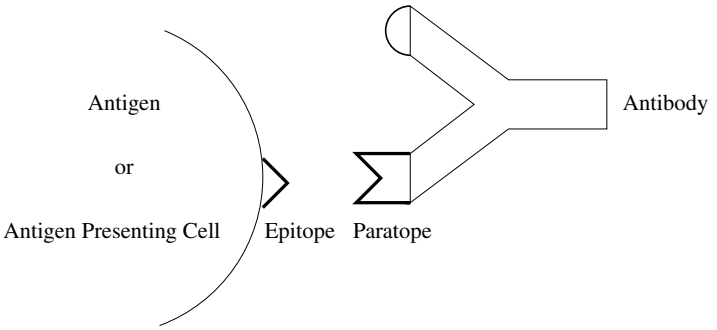

Figure 7 *The epitope of one antigen or an antigen presenting cell is recognized by the paratope of an antibody*

Cytokines are also introduced. Cytokines are groups of proteins secreted by many types of cells. Each cytokine binds to a specific cell's surface receptor signaling a specific action i.e. differentiation into plasma cells, antibody secretion or cell death. They bind also through own receptors constituted from proteins too, see Fig. 8.

B-cells in the immune system secrete antibodies. They also secrete cytokines in order to signal something to another cell. Then, a B-cell will be taken as a generic cell V_i with two components expressed by:

$$V_i = (c_i, P_i) \tag{2}$$

Where:

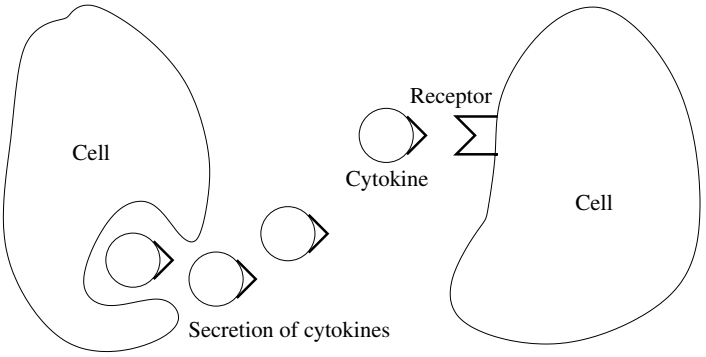

Figure 8 *Cytokines signal the cellular interaction. They are secreted by cells. They are recognized by cell's receptors*

$c_i \in \mathbb{N}$ represents a cytokine. Recovery action to be taken under presence of error.
$P_i \in \mathbb{R}^q = ((p_1)_i, ..., (p_q)_i)$ is a point in a q-dimensional space. P lies within a cube $max\{|\ (p_1)_i\ |, ..., |\ (p_q)_i\ |\} \leq 1$. It represents a protein transformed into the FIN (Formal Immune Network) space. In biological terms it represents an antigen binding site of an antibody or simplifying an antibody. An array containing an input test and its test response, all transformed to the FIN space is an antibody.

In Fig. 9, a two dimensional Formal Immune Network (2D-FIN) is presented. As $q = 2$, each protein has two coordinates in the FIN space.

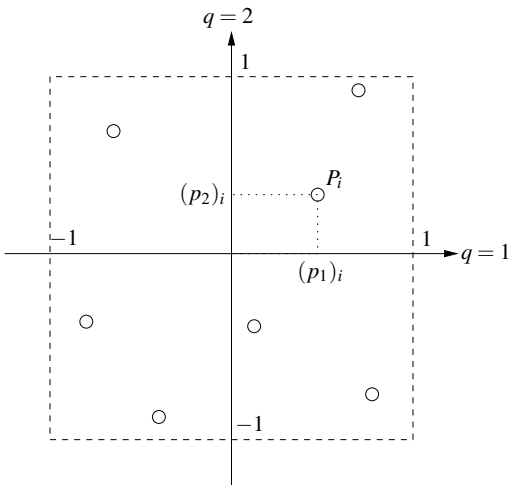

Figure 9 *2D-FIN. Note that q=2 and P represents an antibody (protein) in the FIN space*

5.1 First stage training

Training consist in transforming a given energy matrix A into another antibody matrix P. Matrix A is a composed of arrays of test inputs and its corresponding test response outputs, see Fig. 4. Training is the process of mapping antibodies into the Formal Immune Network space. n dimension training patterns are transformed to reduced dimension patterns (two or three). This takes place by means of Singular Value Decomposition, restricting its terms of decomposition to two or three. Figure 10 introduces the general concept. Singular values and the right singular vectors are used for the calculation of the coordinates of each training pattern into the FIN space. Each point will represent an antibody. In the figure, two outputs are displayed. First, the SVD: singular values, right singular vectors and left singular vectors. Second, the matrix P with the transformed patterns, where each pattern remains linked to its initial c value. The output should be stored in order to be used in the later stages.

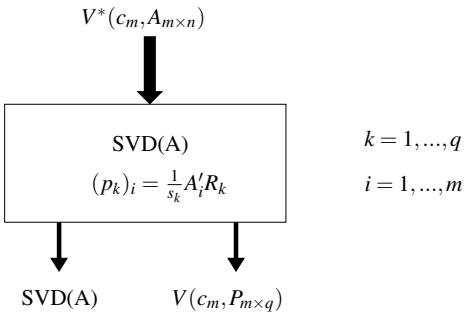

Figure 10 *Training*

A can be written as a linear combination of pairwise orthogonal projections:

$$A = s_1 L_1 R_1' + s_2 L_2 R_2' + s_3 L_3 R_3' + \ldots + s_k L_k R_k' + \ldots + s_r L_r R_r' \tag{3}$$

where:

r rank of the matrix A
L_k left singular vectors
R_k right singular vectors
s_k singular values

Moreover, L_k is of dimension m and R_k of dimension r.

The minimal binding energy is achieved with the pair of proteins whose angles in its spacial configuration form singular vectors. Those singular vectors correspond to the maximal singular values of the matrix A. Singular vectors represent formal protein probes and the singular values their binding energy. As the singular values are ordered in a decreasing order, we can take the first two singular values and its corresponding terms for a 2D-FIN and three terms for a 3D-FIN. In consequence every training vector pattern of dimension n is mapped to only two (or three) values of binding energy in the FIN space [16]. Afterwards, it is necessary to map the training vectors into the FIN space (see Fig. 9) by means of:

$$(p_k)_i = \frac{1}{s_k} A_i' R_k \tag{4}$$

where $i = 1, ..., m$ and $k = 1, ..., q$.

5.2 Second stage training

The *affinity* between two cells V_i and V_j, can be calculated by the distance d_{ij} given by:

$$d_{ij} = max\{|\ (p_1)_i - (p_1)_j\ |, ..., |\ (p_q)_i - (p_q)_j\ |\} \tag{5}$$

Initially given m cells with pairs cytokine-antibody $V_i(c_i, P_i)$, the aim is to reduce similar cells killing one of two cells which distance is less than a given *threshold*. The set of m cells belonging to the *innate immunity* can be represented by a W_0 as shown below:

$$W_0 = \{V_1, ..., V_m\} \tag{6}$$

cFIN means cytokine Formal Immune Network. A cFIN is a set of cells $W \subseteq W_0$. In contrast with a FIN, it considers a second stage training or *maturation*, inspired in the cytokines from the immune system. [14] introduces a two stage second training in order to get a reduced set of cells, therefore improving the resource utilization and the time applied in recognition in the future.

The first stage is *apoptosis* and it intends to reduce the set with the following rule:

If $V_i \in W$ recognizes $V_k \in W$, then remove V_k from cFIN.

Note that *recognition* means:

$$c_i = c_k \tag{7}$$

$$d_{ik} \leq h \tag{8}$$

Where h is a threshold of affinity.

The second stage is *auto-immunization*. It tries to recover accidentally removed cells by the process of apoptosis.

The removed cell V_i nearest to a cell V_k from the set W will be inserted again if $c_i \neq c_k$.

Figure 11 shows graphically the sets and the general concept of the optimization offered by cFIN.

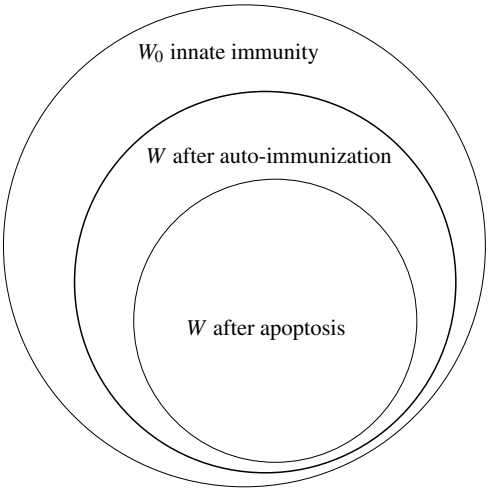

Figure 11 *Sets of cells after Apoptosis and Auto-immunization*

5.3 Recognition

Figure 12 shows an antibody close to an antigen. When the distance between any antibody and the antigen is less than a given *threshold*, recognition in the FIN space is produced.

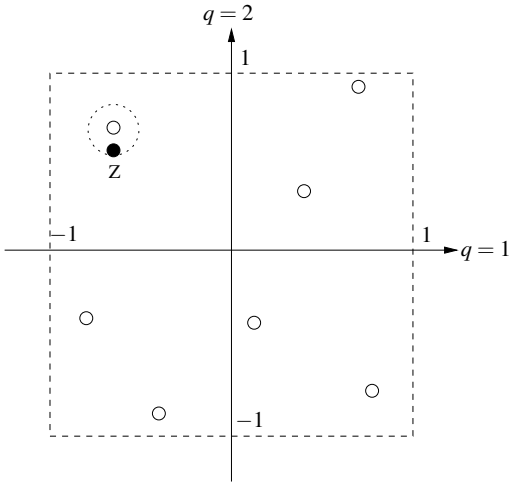

Figure 12 *An antibody recognizes an antigen Z inside its affinity threshold radio*

An antigen $Z = [z_1, z_2, ..., z_n]$ can be seen as an epitope, therefore as a protein, see Fig. 7. An antigen represents the test response linked with its test input. In order to be compared with the antibodies, it should be mapped to a point in the q-dimensional FIN space by:

$$p_k = \frac{1}{s_k} Z' R_k \tag{9}$$

p_k values should be mapped into the FIN space. See also Fig. 13.

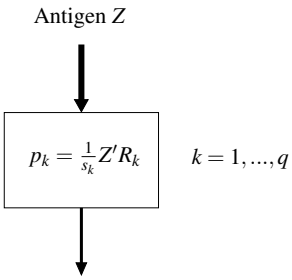

Figure 13 *Recognition*

Cell V_i recognizes antigen Z by assigning it a class c_i, if the distance between the antigen among all antibodies of the cFIN is $d(V_i, P) = min\{d(V_j, P)\}$, for all $V_j \in W$. A test response will be matched with the expected output recognizing whether there is an error or not and applying the determined action signaled by c_i.

For the distance calculation Eq. 5, the Euclidean norm is taken. Nevertheless, the choice of the norm is determined by the appearance of the group of points in the FIN space.

Relating the class c in Fig. 13, it is a numerical value which can also be taken as a symbolic value like ”good”, ”bad”, ”reconfiguration”, etc.

6 Conclusion

Computation times for the training and recognition presented in [13], show that it is feasible to expect a good performance of the model in hardware. Furthermore, the reduced memory constraints obtained with the second training of the cFIN indicates potential towards a distributed error detection and correction scheme. This paper is intended to present a design idea. Currently, an implementation with commercial components and the measure of performance is being carried out.

References

1. Al-Asaad, H., Shringi, M.: On-line built-in self-test for operational faults. In: AUTOTESTCON Proceedings, 2000 IEEE, pp. 168–174. Springer-Verlag New York (2000)
2. Bradley, D., Ortega-Sanchez, C., Tyrrell, A.: Embryonics + immunotronics: A bio-inspired approach to fault tolerance. In: J. Lohn, A. Stoica, D. Keymeulen (eds.) The Second NASA/DoD workshop on Evolvable Hardware, pp. 205–224. IEEE Computer Society, Palo Alto, California (2000)

3. Bradley, D.W., Tyrrell, A.M.: Immunotronics: Hardware fault tolerance inspired by the immune system. In: ICES, pp. 11–20 (2000)
4. Dutt, S., Verma, V., Suthar, V.: Built-in-self-test of fpgas with provable diagnosabilities and high diagnostic coverage with application to online testing. In: Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 27, pp. 309–326. IEEE (2008)
5. Gupta, S.K., Pradhan, D.K.: Utilization of on-line (concurrent) checkers during built-in self-test and vice versa. In: IEEE Transactions on Computers, vol. 45, pp. 63–73. IEEE Computer Society Washington, DC, USA (1996)
6. Irion, A., Kiefer, G., Vranken, H., Wunderlich, H.J.: Circuit partitioning for efficient logic bist synthesis. In: Design, Automation and Test in Europe, 2001. Conference and Exhibition 2001. Proceedings, pp. 86–91 (2001)
7. Kastensmidt, F.L., Carro, L., Reis, R.: Fault-Tolerance Techniques for SRAM-Based FPGAs, *Frontiers in Electronic Testing*, vol. 32. Springer (2006)
8. Kumagai, J.: Swimming to europa. IEEE Spectrum **44**(9) (2007)
9. Rammig, F.J.: Systematischer Entwurf digitaler Systeme. B. G. Teubner Sttutgart (1989)
10. Ratter, D.: Fpgas on mars. XCell journal **50** (2004)
11. Sterpone, L., Violante, M.: A design flow for protecting fpga-based systems against single event upsets. In: Defect and Fault Tolerance in VLSI Systems, 2005. DFT 2005. 20th IEEE International Symposium on, pp. 436–444 (2005)
12. Stroud, C.E.: A Designer's Guide to Built-in Self-Test. Springer (2002)
13. Tarakanov, A.O.: Formal Immune Networks: Self-Organization and Real-World Applications, *Advanced Information and Knowledge Processing*, vol. Part III, pp. 271–290. Springer London (2008)
14. Tarakanov, A.O., Goncharova, L.B., Tarakanov, O.A.: A cytokine formal immune network. In: Advances in Artificial Life, *Lecture Notes in Computer Science*, vol. 3630. Springer Berlin / Heidelberg (2005)
15. Tarakanov, A.O., Kvachev, S.V., Sukhorukov, A.V.: A formal immune network and its implementation for on-line intrusion detection. In: Computer Network Security, *Lecture Notes in Computer Science*, vol. 3685, pp. 394–405. Springer Berlin / Heidelberg (2005)
16. Tarakanov, A.O., Skormin, V.A., Sokolova, S.P.: Immunocomputing, Principles and Applications. Springer New York (2003)
17. Tempesti, G.: A self repairing multiplexer-based fpga inspired by biological processes. Ph.D. thesis, École Polythechnique Fédérale de Lausanne (1998)
18. Tyrrell, A.: Computer know thy self!: A biological way to look at fault tolerance. In: 25th Euromicro Conference (EUROMICRO '99), vol. 2, pp. 21–29. 2nd Euromicro/IEEE Workshop on Dependable Computing Systems (1999)
19. Wang, Z., Chakrabarty, K.: Built-in self-test and defect tolerance in molecular electronics-based nanofabrics. In: Journal of Electronic Testing: Theory and Applications, vol. 23, pp. 145–161. Kluwer Academic Publishers (2007)
20. Wunderlich, H.J.: Test and testable design. In: Architecture design and validation methods, pp. 141–190. Springer-Verlag New York (2000)
21. Xilinx Inc.: Xilinx TMRTool, The First Triple Module Redundancy Development Tool for Reconfigurable FPGAs. URL www.xilinx.com