XIX IMEKO World Congress
Fundamental and Applied Metrology
September 6 11, 2009, Lisbon, Portugal

# ADC FUNCTIONAL TESTING USING ARTIFICIAL IMMUNE SYSTEM

*Cleonilson Protásio de Souza*[1], *Cláudio Leão Torres*[2], *Raimundo C. S. Freire*[3], *Francisco M. de Assis*[4]

[1]Federal Institute of Maranhão, São Luís, Brazil, protasio@ieee.org
[2]Federal Institute of Maranhão, São Luís, Brazil, cltzv@dee.cefet-ma.br
[3]Federal University of Campina Grande, Campina Grande, Brazil, rcsfreire@dee.ufcg.edu.br
[4]Federal University of Campina Grande, Campina Grande, Brazil, fmarcos@dee.ufcg.edu.br

**Abstract** – Artificial immune systems have been considered one of the most promising nature-inspired technique used for novelty detection systems, optimization, identification, etc. One of the main tools of these systems is the Negative-Selection Algorithm that is based on the natural self-nonself discrimination which has made possible the body distinguish any foreign cell from the body's self cell. In this work, it is used the Negative-Selection Algorithm to perform functional testing of analog-to-digital converter. In this way, the evaluation of the converter is performed by a set of immune-based detectors which can detect faulty output responses. Using the immune-based detector set, experimental results have been shown the effectiveness of the proposed method.

**Keywords**: ADC, Functional testing, Artificial immune systems, Negative-Selection Algorithm.

## 1. INTRODUCTION

Nowadays, digital electronic systems are essential for our lives. In this context, a fundamental principle arises: the most the benefits of these systems in our lives, the most is the possibility of causing damage when they fail [1]. Examples come from avionics, medicine, etc. In these areas or others that can cause any damage it is necessary to design high reliability systems [2]. A circuit that is the most common and widely used mixed-signal circuits in electronic systems is the analog-to-digital converter (ADC) [3]. ADCs are widely used because they serve as interface between analog or physical world to the digital logic and subsequently digital processing [4][5]. The importance of ADCs comes from the fact that the performance of the system is directly affected by it [6], i.e., no matter if the system is good but the ADC is faulty. In this way, ADC testing is an important area to increase the system reliability, since this depends on the reliability of its components (ADCs, for examples) [7]. Due to increasing resolution and conversion rates, the challenge and cost of testing ADCs are growing.

On the other hand, natural systems that may be used as a model for error detectors are the biological immune ones which are very complex systems with several mechanisms for defense against pathogenic organisms. The primary purpose of immune systems is to recognize all cells within the body and categorize those cells as body's self cell (*self*) or foreign cell (*nonself*). Such a recognition process is known as the **self-nonself discrimination**. After discrimination, the nonself cells are further categorized in order to induce an appropriate type of defensive mechanism [8]. With the ability to detect nonself, immune systems seem to be an adequate source of inspiration to development of algorithms for early detection of anomalous behavior in systems [9]. Artificial systems coming from immune systems are called artificial immune systems and they are being considered one of the most promising nature-inspired technique used for novelty detection systems [10].

In this work, a functional test of ADC, called immune-ADC, which takes inspiration from the principles of self-nonself discrimination is presented. Using the proposed immune-ADC scheme, the evaluation of the ADC Under Test is performed by a set of pre-computed immune-based detectors which can detect faulty circuit output response.

Using the set of immune-based detectors, the evaluation of the ADC Under Test is performed on-line and the experimental results show the effectiveness of the proposed scheme.

## 2. NEGATIVE-SELECTION ALGORITHM

Basically in biological immune systems, the mechanism for detection of pathogenic organisms consists of lymphocytes that can be thought as detectors which can recognize pathogens and destroy them. This recognition is achieved in part by T lymphocytes (T cells), which have receptors (antibodies) on their surface that can detect foreign proteins (antigens) on pathogenic organisms. To avoid that such systems recognize the body's self cells and only recognize pathogenic organisms, i.e. to perform self-nonself discrimination, a process called negative selection is used. Such process occurs in the thymus where, during the generation of T cells, receptors are made by a random genetic rearrangement process which

ISBN 978-963-88410-0-1 

eliminates T cells that react against self-proteins. In this way, only T cells that do not bind any self-protein are allowed to leave the thymus [8]. These matured T cells then circulate throughout the body to perform immunological functions to protect against pathogenic organisms.

Based on this self-nonself discrimination, Forrest et al. [11] proposed a negative-selection algorithm for change detection. Such an algorithm generates detectors randomly, and eliminating the ones that detect self, so that the remaining generated detectors can detect (probably) any nonself. The negative-selection algorithm is summarized as follows:

- **Definition of the data to be protected.**
  In this phase, it is defined as self the data that need to be protected or monitored. These data can be a collection $R = \{R_1, R_2, \cdots, R_n\}$ of $l$-length strings over a finite alphabet of cardinality $q$. Each of these strings is called self-string.
- **Detector generation phase.**
  Generate candidate detectors randomly and verify if they match any self-string in $R$ (according to a specified matching rule, see Section 2.1). If a match is found, the candidate is rejected. Otherwise, it is accepted as detector in the detector set $D$. This process is repeated until a desired number of detectors have been generated. The generation phase is illustrated in Figure 1.
- **Monitoring phase.**
  Monitor $R$ for changes by continually matching the detectors in $D$ against $R$. If any detector ever matches, then a change is known to have occurred, because the detectors are designed not to match any of the original strings in $R$. This monitoring phase is illustrated in Figure 2.

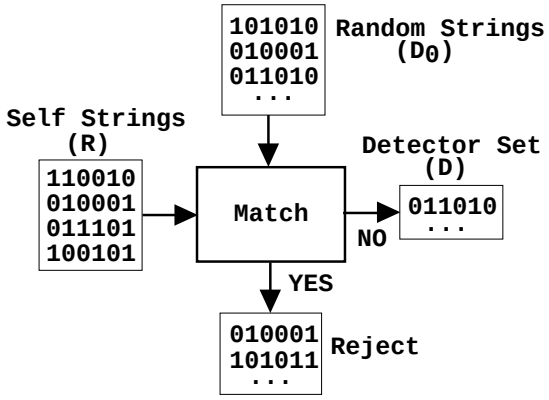


Fig. 1. Negative-Selection Algorithm: detector generation phase. Each detector in $D$ fails to match any string in $R$.

If a change in the protected data was viewed as any string that does not belong to the original data, then this algorithm proposes to generate detectors for (almost) all strings not in the original data and it turns out to be feasible mathematically, that is, a fairly small set of detector strings has a very high probability of noticing a random change to the original data [11]. In addition, it turns out to be robust as it detects (probabilistically) any foreign activity rather than looking for specific known patterns of changes [8].

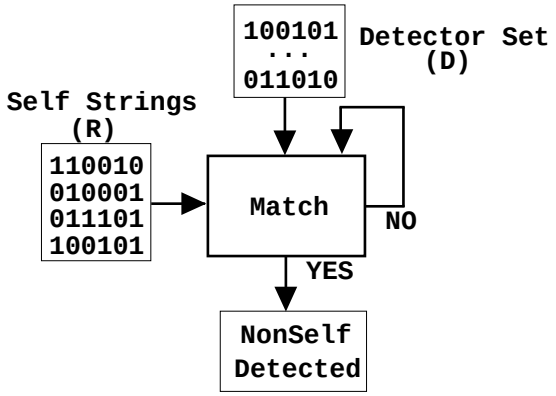


Fig. 2. Negative-Selection Algorithm: monitoring Phase. Monitor $R$ for changes by continually matching the detectors in $D$ against $R$.

The generation phase is the most expensive one since it appears to be computationally difficult to generate valid detectors, which grows exponentially with the size of self collection [11]. However, monitoring phase is the cheapest [11]. In testing area, such facts are advantageous because the generation phase is performed in the design stage and the monitoring phase is performed on-line.

**Example:** suppose that we have eight 4-bit strings to be protected, i.e., our self-string collection is given by

$$R = \{0010,\ 1000,\ 1001,\ 0000,\ 0100,\ 0010,\ 1001,\ 0011\}$$

The generation phase consists in generating random strings ($D_0$), and then matching the strings of $D_0$ against the strings in $R$. Strings from $D_0$ that match a self-string are rejected. Strings that do not match any self-strings become members of the detector collection ($D$).

Suppose that $D_0$ contains the following four random strings:

$$0111\ 1000\ 0101\ 1001$$

Then, $D$ will consist of two strings, 0111 and 0101, the strings 1000 and 1001 will be rejected because each of them matches a string in $R$. In practice, the procedure is to generate random strings sequentially, and to continue generating them until $D$ has a sufficient number of elements.

In monitoring phase, with the collection $D$ of detectors, the state of self can be monitored by continuous matching strings in $R$ against strings in $D$. Suppose that one bit of the last self string (0011) is changed to produce 0111. Then, at some point in the monitoring phase, it would be noticed that the nonself string 0111 matches one of the detector strings (the string 0111), and a change would be reported.

### *2.1 Matching Rules*

In negative-selection algorithm, a criterion of matching is necessary since a perfect match between two strings of equal length $l$ means that at each location in the strings, the symbols are identical. In this case, the matching is completely specific, so a detector will match only a single string. So that detectors could match several strings, i.e., to

increase the capability of detecting, a partial-matching rule is used.

Two partial matching rules are the *r-contiguous* matching rule and the *r-hamming* matching rule. Both the r-contiguous matching rule and the r-hamming matching rule are controlled by the threshold parameter $r$. The higher the value of $r$, the more specific the match is.

A consequence of using a partial matching rule is that there is a trade off between the value of $r$ and the number of necessary detectors to detect nonself strings.

The r-contiguous matching rule was proposed by Forrest et al. [11]. It consists in looking for $r$ contiguous matches between symbols in corresponding positions. Suppose that two strings $x$ and $y$ have symbols from an alphabet of cardinality $q$.

An example based on r-contiguous matching rule is shown in Figure 3.

```
x: 1 1 0 0 1 0 1 0 1 0 1 1 1 0 1 0
y: 0 0 1 0 1 0 1 0 0 1 1 0 1 0 1 1
```

Fig. 3. A match under the r-contiguous matching rule, with strings of length $l = 16$ consisting of symbols from an alphabet with cardinality $m = 2$ with the matching constraint $r = 5$. The strings $x$ and $y$ in the above example will match for all $r \leq 5$.

The r-hamming matching rule is based on Hamming distance and consists in looking for $r$ matches between symbols in corresponding positions (not necessary contiguous).

An example based on r-hamming matching rule is shown in Figure 4.

```
x: 1 1 0 0 1 0 1 0 1 0 1 1 1 0 1 0
y: 0 0 1 0 1 0 1 0 0 1 1 0 1 0 1 1
```

Fig. 4. Matching under the r-hamming matching rule, with strings of length $l = 16$ consisting of symbols from an alphabet with cardinality $m = 2$ with the matching constraint $r = 9$. The strings $x$ and $y$ in the above example will match for all $r \leq 9$.

For r-contiguous matching rule, the probability of two random strings match in at least $r$ contiguous positions is given by:

$$P_{M_r^c} \approx q^{-r}\left[\frac{(l-r)\,(q-1)}{q} + 1\right] \tag{1}$$

for $q^{-r} \ll 1$ [12].

For r-hamming matching rule, the probability of two random strings match in at least $r$ positions is given by:

$$P_{M_r^h} = q^{-l}\left(\sum_{i=r}^{l} \binom{l}{i} (q-1)^{l-i}\right) \tag{2}$$

As the probability of one detector does not match a nonself-string is $1-P_{M_r}$, and the probability of two detectors does not match a nonself string is $(1-P_{M_r})^2$, and so on, then, the probability of $h$ detectors fail to detect a nonself string, i. e., the probability of detection error $P_e$ is given by:

$$P_e = (1 - P_{M_r})^h \tag{3}$$

## 3. PROPOSED ADC FUNCTIONAL TESTING

Based on the negative-selection algorithm, this work proposed a functional testing scheme of ADC using an immune approach. Figure 5 shows the primary scheme comprised of an ADC Under Test with resolution of *n*-bits and an ADC of reference with lower resolution *m*-bits where $m < n$. $V_i$ is a ramp signal that is used for stimulating both ADCs. The binary responses of the ADCs form the vector

$$\dot{R}^i = \{R_1, R_2, \cdots, R_{n-1}, R_n, R_{n+1}, R_{n+2}, \cdots, R_{n+m}\}$$

where $0 \leq i \leq 2^n$.

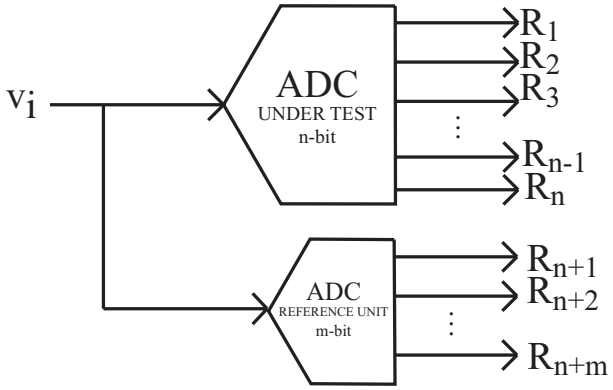


Fig. 5. Primary scheme of the proposed immune-based ADC functional testing scheme.

Considering the ADCs are fault-free and applying the ramp signal, it is obtained $2^n$ digital output vectors $\dot{R}^i$ or

$$\dot{R} = \dot{R}^1, \dot{R}^2, \cdots, \dot{R}^{2^n}.$$

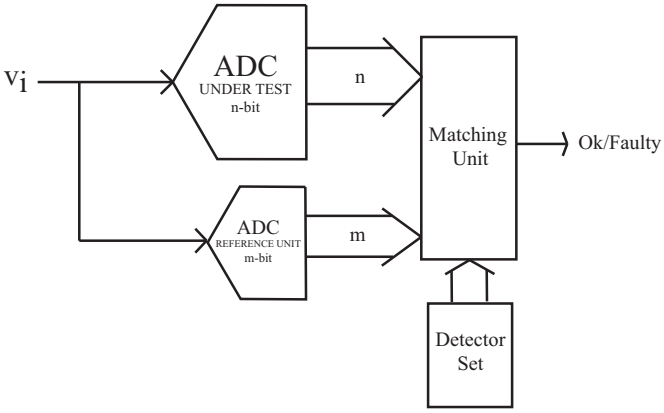


Fig. 6. Complete scheme of the proposed immune-based ADC functional testing scheme.

Let $\dot{R}$ be the the data to be protected using NSA. After this, it is generated a set of detectors $D$ using the NSA considering the fault-free ADCs responses as the self-strings. Using the complete scheme shown in Figure 6 that emulates the monitoring phase of NSA, it is possible to test ADC Under test since if any binary response matches a detector,

Table 1. Experimental results ($d$: number of detectors; $c$: matching rule; $r$: parameter; $F_i$: fault).

| **Detected (Y = yes, N = No)** | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $d$ | 20 | | | | | | 80 | | | | | | 320 | | | | | |
| $c$ | r-Hamming | | | r-contiguous | | | r-Hamming | | | r-contiguous | | | r-Hamming | | | r-contiguous | | |
| $r$ | 13 | 14 | 15 | 13 | 14 | 15 | 13 | 14 | 15 | 13 | 14 | 15 | 13 | 14 | 15 | 13 | 14 | 15 |
| $F_1$ | Y | Y | N | N | N | N | Y | N | Y | Y | N | Y | Y | N | Y | Y | N | Y |
| $F_2$ | Y | Y | N | N | N | N | Y | Y | Y | Y | N | Y | Y | Y | Y | Y | N | Y |
| $F_3$ | Y | N | N | N | N | N | Y | Y | Y | Y | N | Y | Y | Y | Y | Y | N | Y |
| $F_4$ | Y | Y | N | N | N | N | Y | Y | N | Y | Y | Y | Y | Y | Y | Y | Y | Y |
| $F_5$ | Y | N | N | N | N | N | Y | Y | N | N | N | N | Y | Y | Y | Y | Y | Y |
| $F_6$ | Y | N | N | Y | N | N | Y | Y | N | Y | N | N | Y | Y | Y | Y | Y | Y |
| $F_7$ | Y | Y | N | Y | N | N | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y |
| $F_8$ | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y |

which is sequentially generated, then the ADC is considered faulty. Otherwise it is considered fault-free.

The general procedures for designing the proposed immune-based ADC functional testing scheme are as follows.

1) Consider the $2^n$ $n+m$-bit responses $\dot{R} = \dot{R}^1, \dot{R}^2, \cdots, \dot{R}^{2^n}$ when the ramp signal is applied to the scheme and consider the ADC fault-free.
2) Consider each fault-free response $\dot{R}^i$ (where $R_i \in \{0,1\}^{n+m}$ and $1 \leq i \leq 2^n$) as a self-string to be protected. In this way, the self data are formed by $2^n$ binary vectors of length $n+m$.
3) Apply the self-strings in the negative-selection algorithm and generate a set of detectors $D = \{D_1, D_2, \cdots, D_h\}$ where $h$ is the number of detectors obtained.
4) Use the obtained detectors to test the ADC using the scheme shown in Figure 6.
5) If a detector matches with an output response, then it is known that the ADC is faulty.

## 4. EXPERIMENTAL RESULTS

The experimental simulations were implemented assuming single stuck-at fault model at digital part of ADC under test. The ADC under test was a 12-bit Ladder one and the ADC reference unit was a 3-bit flash one. Table 1 shows the obtained experimental results considering 8 faults ($F_1$ to $F_8$). Row $d$ shows the number of detectores generated in NSA. Row $c$ shows the matching rule used and row $r$ the parameter. For instance, in Row $F_1$, the fault $F_1$ was detected using 20 detectors and considering r-Hamming and $r = 13$, but it was not detected using r-contiguous under the same $r$. For the remaining faults, Table 1 shows the obtained results.

It is important to note that all considered faults was detected using 20 detectors and considering r-Hamming and $r = 13$ and that it is not necessary any specific input stimulus. The ADC testing is really performed on-line.

## 5. CONCLUSION

In this work was presented the use of Negative-Selection Algorithm to perform functional testing of analog-to-digital converter. The ADC evaluation is performed by a set of immune-based detectors which can detect faulty output responses. Using the proposed method and based on preliminary experimental results, it was shown that the proposed method was able to detect faulty ADCs indicating the effectiveness of the method.

## ACKNOWLEDGMENTS

The authors would like to thank the financial support provided by the Brazilian National Council for Scientific and Technological Development (CNPq) and by Maranhão State Research Supporting Foundation (FAPEMA).

## REFERENCES

[1] Avizienis, A. Toward Systematic Design of Fault-Tolerant Systems. *IEEE Transactions on Computer*, 30:51–58, Apr 1997.
[2] Ali, L. at al. Challenges and directions for testing IC. *Integration, the VLSI Journal*, 37:17–28, February 2004.
[3] International technology roadmap for semiconductors. 2001.
[4] Wen, Y. C. and Lee, K. J. An on Chip ADC test structure. *Proceedings of Design, Automation and Test in Europe (DATE '00)*, page 2000.
[5] Walden, R. H. Analog-to-digital converter survey and analysis. *IEEE Journal on Selected Areas in Communications*, 17(4):539–550, April 1999.
[6] Serra, A. C.; Alegria, F.; Martins, R. and Fonseca, M. Analog to digital converters testing - new proposals. *Computer Standards & Interfaces*, 26(1):3–13, January 2004.
[7] Jalote, P. *Fault Tolerance in Distributed System*. Prentice Hall, Inc, Englewood Cliffs, New Jersey, 1994.
[8] Dasgupta, D., and Attoh-Okine, N. Immunity-based system: a survey. *IEEE International Conference on Computational Cybernetics and Simulation*, pages 369–374, 1997.
[9] Costa Branco, P. J., Dente, J. A., and Vilela Mendes, R. Using Immunology principles for fault detection. *IEEE Transaction on Industrial Electronics*, 30(2):302–375, 2003.
[10] Seredynski, F. and Bouvry, P. Some Issues in Solving the Anomaly Detection Problem using Immunological Approach. *Proceedings on 19th IEEE International Parallel and Distributed Processing Symposium*, pages 188–196, 04-08 April 2005.
[11] Forrest, S. at al. Self-Nonself discrimination in a computer. *Proceedings of IEEE Symp. on Research in Security and Privacy*, pages 202–212, May 1994.
[12] Bradley, D.W. and Tyrrell, A.M. Immunotronics - Novel Finite-State-Machine Architectures with Built-In Self-Test Using Self-Nonself Differentiation. *IEEE Transactions on Evolutionary Computation*, 6:227–238, Jun 2002.