XVIII IMEKO WORLD CONGRESS
Metrology for a Sustainable Development
September, 17 - 22, 2006, Rio de Janeiro, Brazil

TESTING VLSI CIRCUIT USING ARTIFICIAL IMMUNE SYSTEM

C. P. Souza[1], *R. C. S. Freire*[2] *and F. M. Assis*[3]

[1]Federal University of Campina Grande, Brazil, protasio@dee.ufcg.edu.br
[2]Federal University of Campina Grande, Brazil, rcsfreire@dee.ufcg.edu.br
[3]Federal University of Campina Grande, Brazil, fmarcos@dee.ufcg.edu.br

Abstract: A VLSI circuit test scheme taking inspiration from the Human Immune System is presented. Such a scheme is based on the Negative-Selection Mechanism which provides the human body with the capability to discriminate between the self (body's own cell) and any foreign cell (non-self). Based on this, it is design a output response analyzer which is able to evaluate is the circuit is faulty. Experimental results showing the effectively of the proposed scheme are presented.

Keywords: VLSI circuit testing, output response analyzer, artificial immune system.

1. INTRODUCTION

A natural system that may be used as model to an error detector is the human immune system.The main purpose of the human immune system is to recognize all cells within the body and categorize those cells as *self* or *nonself*. With its ability to detect *nonself*, the human immune system seems to be an adequate source of inspiration to development of algorithms for early detection of anomalous behavior in systems [1]. Forrest et al. [2] proposed a Negative-Selection Algorithm to try to solve the problem of ensuring the security of computer systems.

On the other hand, Built-in self-test (BIST) is a promising VLSI test technique which requires to incorporate a test pattern generator (TPG) and an output response analyzer (ORA) into the chip itself and then both test patterns generation and output data evaluation are performed on-chip hardware.

The TPG is used to apply a test vector sequence $T = \{T_n, \cdots, T_2, T_1\}$ to the circuit under test (CUT) as shown in Figure 1. The test response sequence, $R = \{R_n, \cdots, R_2, R_1\}$, is applied into the ORA [3]. Generally, the fundamental block of the ORA is a compressor, which compresses R into a compact *signature*. The evaluation consists in comparing the obtained signature with a pre-determined fault-free signature at the end of test. This comparison determines whether the CUT is fault-free or not. Due to the loss of information during compression, the CUT might be declared fault-free although the CUT is actually faulty. Such fail on the CUT evaluation is called *aliasing*.

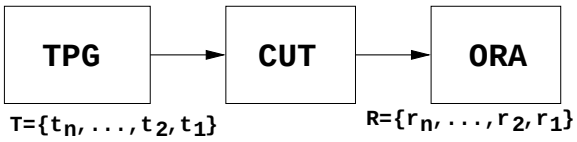

Fig. 1. BIST environment.

In this work, an Immune-Based ORA based on the application of the Negative-Selection Algorithm coming from the Human Immune System is presented. Some experimental results showing the effectively of the proposed scheme are presented.

2. NEGATIVE-SELECTION ALGORITHM

The Negative-Selection Algorithm is based on the principles of *self-nonself* discrimination of the human immune systems. This discrimination is achieved in part by T-cells, which have receptors on their surface that can detect foreign proteins (antigens). During the generation of T-cells, receptors are made by a pseudo-random genetic rearrangement process. Then, they undergo a censoring process, called negative selection, in the *Thymus* where T-cells that react against self-proteins are destroyed, so only those that do not bind to self-proteins are allowed to leave the *Thymus*. These matured T-cells then circulate throughout the body to perform immunological functions to protect against foreign antigens. The Negative-Selection Algorithm works on similar principles, generating detectors randomly, and eliminating the ones that detect *self*, so that the remaining T-cells can detect any *nonself* [4]. This algorithm approach may be summarized as follows:

Define Protected Data

Define *self* as a collection $R = \{R_n, \cdots, R_2, R_1\}$ of strings of length l over a finite alphabet (a collection that needs to be protected or monitored).

Generation of Detectors

Generate a set D of *detectors* from a set D_0 of candidates. Each detector of D fails to match any string in R. Instead of exact or perfect matching, the method uses a partial matching rule. This generation is illustrated in Figure 2. The candidate detectors are generated randomly and then tested (censored) to see if they match any self string. If a match is found, the candidate is rejected. This process is repeated until a desired number of detectors are generated.

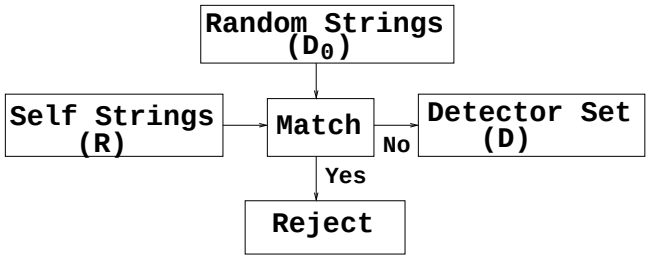

Fig. 2. Negative-Selection Algorithm: Generation of Detectors.

Monitoring

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 to not match any of the original strings in R. This monitoring phase is illustrated in Figure 3.

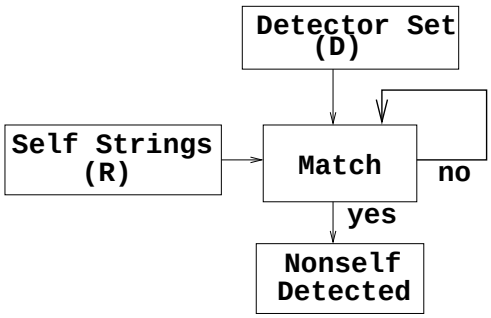

Fig. 3. Negative-Selection Algorithm: Monitoring.

If we view the data set being protected as a set of strings over a finite alphabet, and a change to that data as any string not in the original set, then this algorithm proposes to generate detectors for (almost) all strings not in the original data set 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 [2].

Example

Suppose that we have eight 4-bit strings to be protected, i.e, our self string collection:

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

The *detector generation* consists in generating random strings (set 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 set (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 being rejected because each of then match a string in R.

At *monitoring phase*, with this collection D of detectors, the state of *self* can be monitored by continually 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 this 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.

Partial matching

In Negative-Selection Algorithm, a criterion of partial matching is necessary to perform the censoring and monitoring phase since a perfect match between two strings of equal length l means that at each location in the string, the symbols are identical. In this case, the matching is completely specific, so the detector will detect only a single string. Moreover, perfect matching is extremely rare between strings of any reasonable length [2] [5].

A matching rule, called **r-contiguous** [2], consist in looking for r contiguous matches between symbols in corresponding positions. Suppose that two strings x and y have symbols in a defined finite alphabet. Then, we say that they match if x and y agree (match) in at least r contiguous positions. An example based on r-contiguous match rule is shown in Figure 4.

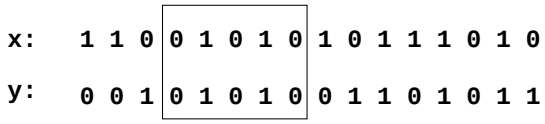

Fig. 4. A match under the r-contiguous match rule, with 16-bits strings with the matching constraint $r = 5$. The strings x and y match for all $r \leq 5$.

Another matching rule, called **r-hamming**, is based on Hamming distance and consists in looking for r not-necessarily contiguous matches between symbols in corresponding positions. Then, we say that they match if x and y agree (match) in at least r positions. An example based on r-hamming match rule is shown in Figure 5.

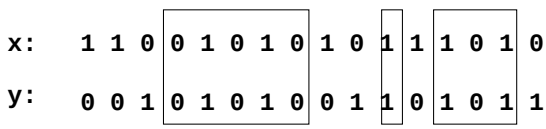

Fig. 5. A match under the r-hamming match rule, with 16-bits strings with the matching constraint $r = 9$. The strings x and y match for all $r \leq 9$.

TABLE I

MINIMAL NUMBER OF DETECTORS TO ATTAIN ZERO-ERROR DETECTION (ZERO ALIASING)

Circuit	Input Lines	Output Lines	Faults	Testing Length	r-contiguous	r-hamming
c1355	41	32	1574	84	17	9
c1908	33	25	1879	112	19	11
c2670	233	140	2747	100	13	7
c3540	50	22	3428	141	21	12
c5315	178	123	5350	106	17	6
c6288	32	32	7744	27	11	4
c7552	207	108	7550	198	24	11

3. PROPOSED IMMUNE-BASED OUTPUT RESPONSE ANALYZER

The main contribution oh this work is the design of a new ORA scheme based on the Negative-Selection Algorithm. Such a scheme, called Immune-Based ORA, can be seen in Figure 6. If a test pattern sequence $T = \{T_n, \cdots, T_2, T_1\}$ is applied to the inputs of the fault-free circuit then a fault-free test response sequence $R = \{R_n, \cdots, R_2, R_1\}$, where $R_i \in \{0,1\}^q$ and q is the circuit output number, is obtained.

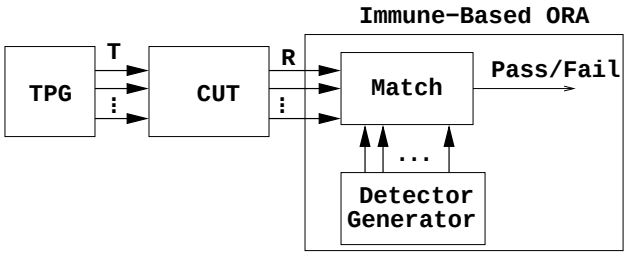

Fig. 6. Proposed Immune-Based ORA.

Considering the elements of R as the elements to be protected (self strings) and applying the Negative-Selection Algorithm, a testing of the CUT can be made. The procedure are the following. First, obtain the fault-free test response sequence $R = \{R_n, \cdots, R_2, R_1\}$. Next, apply these self strings in the Negative-Selection Algorithm to generate N_D detector strings using a given matching rule. Finally, an evaluation of the CUT can be made using these detector strings using the scheme of Figure 6 to performs the *monitoring phase*.

4. EXPERIMENTAL RESULTS

The experimental results were obtained on some *ISCAS'85* benchmark circuits. Table I shows the circuit name, input number, output number and fault number in Column 1, 2, 3 and 4, respectively. In Column 6 and 7, it is reported the minimal number of detectors in order to attain zero-error detection, i.e., zero aliasing, using r-contiguous and r-hamming, respectively, and a suitable matching constraint r. In Column 5, the test length is given. In all cases, it was necessary less detectors using *r-hamming* than using *r-contiguous*. This occurs because the *r-hamming* matching rule covers better the space of nonself than the *r-contiguous* matching rule.

5. CONCLUSIONS

In this work, an ORA scheme based on Human Immune System for using in BIST environment is presented. With the application of the Negative-Selection Algorithm stemmed from Immune System, the proposed immune-based ORA has been capable to discriminate between the fault-free response (*self*) and any fault response (*nonself*). Simulation results have shown that zero-error detection can be achieved according to the obtained number of detectors.

REFERENCES

[1] Costa Branco, P. J., Dente, J. A., and Vilela Mendes, R., "Using Immunology principles for fault detection," *IEEE Transation on Industrial Electronics*, vol. 30, no. 2, pp. 302–375, 2003.

[2] Forrest, S., Perelson, A. S., Allen, L and Cherukuri, R., "Self-Nonself discrimination in a computer," *Proceedings of IEEE Symposium on Research in Security and Privacy*, pp. 202–212, May 1994.

[3] Ali, L.; Sidek, R.; Aris, I.; Suparjo, B. S. and Ali, A. M., "Challenges and directions for testing IC," *Integration, the VLSI Journal*, vol. 37, pp. 17–28, February 2004.

[4] Dasgupta, D., and Attoh-Okine, N., "Immunity-based system: a survey," *IEEE International Conference on Computational Cybernetics and Simulation*, pp. 369–374, 1997.

[5] Bradley, D.W.; Tyrrell, A.M., "Immunotronics - novel finite-state-machine architectures with built-in self-test using self-nonself differentiation," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 227–238, Jun 2002.