Australian Journal of Basic and Applied Sciences, 2(4): 920-928, 2008
ISSN 1991-8178

Using Immune Genetic Algorithm in ATPG

Mehdi Azimipour, Mohammad Reza Bonyadi and Mohammad Eshghi

Department of Electrical and Computer Engineering, Shahid Beheshti University, Evin, Tehran, Iran

Abstract: In this paper, an immune genetic based algorithm (IGA) for random test pattern generation was proposed. Genetic algorithms (GA) solve many search and optimization problems, effectively. However, they may drop into local optimal solutions; or they may find the optimal solution by low convergence speed. To overcome these problems, we used the immune concept and GA algorithm for random-based test generation. In the proposed algorithm, some of the main characteristics of the immune system were used to enhance the GA algorithm. As a result, a new random-based test pattern generation technique based on immune genetic algorithm (IGA) was presented. Experimental results showed that the proposed algorithm improved the ability of global search, avoided dropping into the local optimal solutions and increased the speed of computation convergence with respect to previously proposed non-immune GA algorithms. The proposed algorithm improved the test size with a factor of about 25 % in comparison with non-immune algorithms.

Key words: Automatic Test Pattern Generation (ATPG), Genetic algorithm, Immune Genetic algorithm, Affinity.

INTRODUCTION

Today, digital systems are extremely intricate and ever increasing in complexity, which are required for use in a widening range of domestic and industrial applications (Breuer and Friedman, 1997). The systems in which complex digital integrated circuits are used are medical, military or flight control systems, which all heavily depend to their correct and reliable operation. In order to ensure reliability of these circuits, it is necessary to test their performance to identify any defects prior to using them in a fully operational environment (Arslan and O'Dare, 1997).

Pre-designed functional blocks which called intellectual property (IP) cores, are being widely used for reducing cost and time-to-market of system-on-chips (SOC). These cores must be tested by precomputed test sets which provided by the core vendors. The test vectors are generated by efficient automatic test pattern generator (ATPG) tools. The generation of test patterns with high fault coverage rate is a very expensive process for large circuits with high fault coverage rate. An efficient ATPG tool reduces the test pattern generation time and cost, beside the high fault coverage rate. Recently some scan-based design-for-testability (DFT) approaches (Azimipour and Eshghi, 2008; Azimipour, *et al*. 2007; Al-Yamani, *et al*. 2005) are recently proposed to apply pre-computed test vectors to the system-on-chip to reduce the test time and cost during test process.

Test generation problem can be viewed as a finite space search problem to find appropriate logic assignment to the primary inputs such that given faults detected (Goel, 1981). The size of the search space is exponential to the number of primary inputs; and test generation problem is proven to be an NP-complete (Abramovici, *et al*. 1994).

In deterministic approach, in order to detect a target fault, the fault must be excited and the corresponding fault effects propagated to a primary output. Backtracing is an essential step in deterministic algorithms. Test generation using deterministic, fault-oriented algorithms is highly complex and time consuming. New approaches are needed to augment the existing techniques, both to reduce execution time and to improve fault coverage. Fault simulators are used extensively in the design of electronic circuits. The first simulation-based test generator proposed by Seshu and Freeman (Seshu and Freeman, 1962). Since then, several simulation-based test generators have been developed to simplify test generation process (O'Dare and Arslan, 1994; Ivask, *et al*. 1998; Rudnick, *et al*. 1997; Rad and Eshgh, 2007). In a simulation-based approach, processing occurs in the forward direction only and no backtracing is required. Therefore, complex component types are handled more easily.

Corresponding Author: Mehdi Azimipour, Department of electrical and computer engineering, Shahid Beheshti University, Evin, Tehran, Iran
Tel: +989112742543 E-mail: mehdi.azimipour@gmail.com E-mail: mehdi.azimipour@gmail.com

GA described by Goldberg (Goldberg, 1989) is composed of chromosomes and three evolutionary operators: selection, crossover and mutation. Genetic algorithm is a random search which is specially suited to solve large scale combination optimization problem. Genetic algorithms have been successfully applied in different areas of VLSI design, especially in test branches such as test pattern generation (O'Dare and Arslan, 1994; Ivask, *et al.* 1998; Rudnick, *et al.* 1997; Rad and Eshgh, 2007). Using genetic algorithm in test pattern generation is effective in achieving a compact test set with high fault coverage and reducing the generation time. GA is used for simulation-based test generation in (Seshu and Freeman, 1962).

Immune genetic algorithm (IGA) proposed by Jiao and Wang in 2000. Theoretically, the aim of using immune concept with GA is to utilize the locally characteristic information for seeking the ways and means of finding the optimal solution when dealing with difficult problems (Jiao and Wang, 2000). To be exact, it utilizes the local information to intervene in the globally parallel process and avoids repetitive and useless work during the searching process, in order to overcome the blindness in the crossover and mutation actions. According to (Jiao and Wang, 2000), IGA can improve the global exploration performance of GA by combining the immune theory from biology. The Jiao and Wang (2000) theoretical analysis and simulations show that IGA is not only feasible, but also effective to alleviating the degeneration phenomenon in the original GA, thus greatly increasing the converging speed. Genetic algorithm-based test generation is a class of simulation-based test generation with some shortcomings. It is easy to drop into local optimal solution or find the solution by low convergence speed.

To overcome these problems, we use the immune concept and GA algorithm for random-based test generation. We use some of the main characteristics of the immune system to enhance the GA. As a result, a new random-based test pattern generation technique based on immune genetic algorithm (IGA) is presented. Our experiments show that this algorithm increases the fault detection rate while decreases test size and computational time for the same predefined coverage limits.

MATERIAL AND METHODS

Automatic test pattern Generation: Increasing complexity of VLSI circuits has led to a progressive need for efficient test generation method that ensures a fault-free performance for the circuit under test. The efficiency of these methods is measured using the fault coverage, computational time and the length of the test set.

ATPG for a given target fault consists of two phases: the fault activation phase and the fault propagation phase. Breuer (Breuer, 1971) uses a fault simulator to evaluate sets of random vectors and to select the best vector to apply in each time frame. Weighted random pattern generators are interfaced with fault simulators in (Schnurmann, *et al.* 1975) and high coverage are obtained for combinational circuits. The test generator in (Snethen, 1977) is built around fault simulator; the next candidate vector is generated using the hamming distance with the previous test vector. The CONTEST test generator (Agrawal, *et al.* 1989) also uses Hamming distance to generate the candidate test vectors; the test vector selection is based on cost function calculated during concurrent fault simulation.

Genetic approach for test pattern generation:

Test pattern generation using deterministic, fault-oriented algorithms is highly complex and time consuming. New approaches are needed to augment the existing techniques, both to reduce execution time and to improve fault coverage. Genetic algorithms (GAs) solve many search and optimization problems, effectively. Since test generation is also a search process over a large vector space, GA is an ideal candidate for this problem.

GA was first used for simulation-based test generation in (Seshu and Freeman, 1962). Several approaches to test generation using GAs have been proposed in (O'Dare and Arslan, 1994; Ivask, *et al.* 1998; Rudnick, *et al.* 1997; Rad and Eshgh, 2007). In (O'Dare and Arslan, 1994; Ivask, *et al.* 1998; Rudnick, *et al.* 1997; Rad and Eshgh, 2007), the fitness evaluation and population scoring is low-cost and only based on the fault coverage of each test vector. The disadvantage of this technique is that if a dropping fault simulation is used, experimentally after almost 10 generations, the generated vectors stop detecting remaining faults. The reason is that after almost 10 generations, all individuals are similar to those whose faults are dropped. The authors in (Rudnick, *et al.* 1997) use more complex and expensive computational for fitness function to eliminate this similarity. This method has resulted in better final test set, but it is very expensive; because of using expensive simulators and computing complex fitness function. A new operator named revocation is used in (Rad and Eshgh, 2007) for canceling this similarity. In this method, after each generation, the best vector in population is put on the final test set and then rescored with a new decreasing fitness. Experimental results show that this method have the small improvement over (O'Dare and Arslan, 1994) and (Ivask, *et al.* 1998).

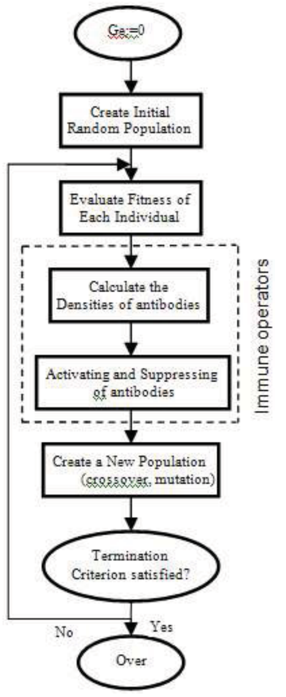
Gen=0
Create Initial Random Population
Evaluate Fitness of Each Individual
Calculate the Densities of antibodies
Activating and Suppressing of antibodies
Immune operators
Create a New Population (crossover, mutation)
Termination Criterion satisfied?
No
Yes
Over

Fig. 1: Flow chart of the proposed IGA

The Immune system Simões, et al., 2003)

The immune system (IS) is a complex, distributed and multi-layered system, which includes cells, molecules and organs that contribute an identification mechanism capable of recognize and eliminate foreign molecules called antigens. When antigens invade, the antibodies that can match these antigens are activated and generate many new antibodies to restrain the antigens. Then the immune system reaches a new balanceable state.

The human body contains a large number of immune cells. Some belong to innate IS, e.g. the macrophages, while others are part of the acquired, or adaptive IS and are called lymphocytes. There are mainly two types of lymphocytes, the B-cells and the T-cells, which cooperate but play different roles in the immune response. The B-cells can be further decomposed into plasma B-cells and memory B-cells. The same happens with the T-cells, which can be partitioned into helper T-cells and killer T-cells. The main functions of B-cells are the production and secretion of antibodies as a response to exogenous organisms. Each B-cell produces a specific antibody, which can recognize and bind to specific pathogen. In order to do their job correctly, the B-cells replicates by a process called clonal selection. This process is similar to the evolution of a population by means of a genetic algorithm using only mutation, which only those cells that have high affinity with antigen proliferate. Therefore, the B-cells that has antibodies, which bind to the pathogen, are selected and cloned. Nevertheless, during cloning some variations may occur due to a process of somatic hypermutation.

This may increase the affinity between the antibodies an antigen, making the B-cells more adapted to bind the antigen. When an antigen enters an organism for the first time, only a few numbers of B-cells can recognize it. Those cells are then simulated to produce antibodies specific to the antigen (primary response). But, the IS can remember patterns that have been seen previously. This is possible because some cells, including the B-cells, become memory cells, which persist in the circulation and are capable of recognizing enemies when and if they attack again. Therefore, the next time the same pathogen invades the organism, the response of the IS will be much faster and stronger (secondary response). It may say that the IS has learning capabilities based on the memory cells.

If each antibody recognize only a specific antigen, how is possible that a huge number of pathogen can be recognized by the IS? The answer to this question remains in the diversity of antibodies that avoid generation of too many similar antibodies which match well the objective function. Balance mechanism of immune system, with activating and suppressing antibodies, can generate necessary antibodies in proper quantity. So, it improves the ability of global search, avoid drop into the local optimal solution and increases the speed of computation convergence.

Table 1: Population size for different vector length

Population Size	Vector Length
8	<4
16	4-16
16	17-49
24	50-63
24	64-99
32	>99

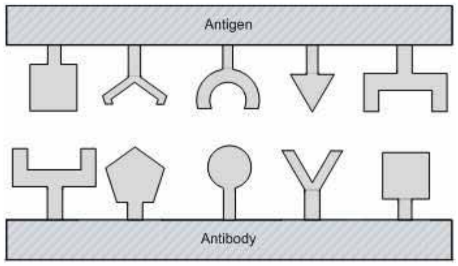

Fig. 2: An antigen binding to a lymphocyte with an exact match

Mapping between the immune system and an optimization problem is done as follows. The immune response represents solutions and antigens represent the problem to solve. In other words, the optimization problem is described by an environment of antigens. The positive/negative selection mechanism is used to control the agent proliferation by eliminating useless or bad solutions.

***Using Immune Genetic Algorithm in ATPG*:**

In this paper, an immune genetic algorithm approach to ATPG for VLSI circuits is introduced. The immune genetic algorithm adds an immune operation to the genetic algorithm. The purpose of the immune operation is to avoid locally optimal solutions, or to avoid finding an optimal solution with a low convergence speed. Individuals are considered to be antibodies (analogous with the biological immune system) and the immune operation consists of two steps: calculating antibody density and activating and suppressing antibodies. Antibody density is the density of similar individuals in the population. Similarity between individuals is the affinity, which is based on Hamming distance. Activating and suppressing antibodies involves adjusting the selection probability of an antibody (individual) for reproduction and is based on both its fitness (fitness is based on how many faults the individual detects) and the antibody's density. The effect of either higher fitness or higher density, increases the probability of selection and the effect or either lower fitness or Lower density, decreases the probability. Results show both a reduction in test sizes and an improvement in execution times compared with other results for combinational benchmark circuits.

This immune concept tries to keep the population immune from local optima by increasing the chances of higher density individuals for selection and suppressing chances of lower density individual. The pseudo code of proposed IGA algorithm for ATPG is shown in Fig.1. Immune operators are showed in dashed section. The description of each step of the proposed algorithm is as follows:

Generation the initial antibodies:

In this step, the antibodies (test vectors) are created randomly on feasible space. Population size should be large enough in order to ensure adequate diversity; however, it is a trade off between getting higher convergence rate with larger search space and less genetic operation time. Population size in algorithm proposed in (Ivask, *et al.* 1998) is constant value for all circuits. Experiments have proven that required population size increases with increasing test vector length. In this paper, we use a more exact population size used in (Rad *et al.,* 2007) which is shown in Table 1.

Calculating the fitness:

The fitness function provides a quantification of the quality of the chromosome. It is the fitness of a chromosome that determines whether the chromosome will be selected to produce offspring and quantifies its chance for survival among the other chromosomes in the population to the next generation.

The fitness function is problem-specific. In this paper fault simulation with fault dropping is used in order to evaluate the test vectors. The score giving to each individual is equal to number of faults it detects. The fitness function for given test vector is calculated by Eq.1:

$$F(x)=\frac{Num.of\ \text{detected}\ faults\ by\ test\ vector(x)}{Total\ number\ of\ faults}$$

Calculation of affinities:

Affinity refers to the degree of binding of the cell receptor (Lymphocytes) with the antigen. Lymphocytes are able to recognize non-self (i.e. antigens) by binding to them with chemical. To ensure that immune system can recognize as many antigens as possible, exact match is not required. A match occurs if a given number of contiguous features complement each others. The number of features required to bind, before a match can be made, is known as affinity threshold. The higher the affinity, the stronger the binding and as a result, better the immune recognition and response. In this paper, the degree of similarity between two vectors is considered as the affinity.

As the antibody (Ab) and antigen (Ag) affinity is related to their distance, it can be estimated via any distance measure between two strings (or vectors), such as the Euclidean, the Manhattan, or the hamming distance. Hence, if the coordinates of an antibody are given by Ab=(Ab_1, Ab_2, …, Ab_l) and those of an antigen are given by Ag=(Ag_1, Ag_2, …, Ag_l), then the distance λ between these two vectors could be defined as Eq. 2-4 (Castro, *et al.* 2003):

$$\lambda(Ab,Ag)=\sqrt{\sum_{i=1}^{L}(Ab_i-Ag_i)^2},$$

$$\lambda(Ab,Ag)=\sum_{i=1}^{L}|Ab_i-Ag_i|,$$

$$\lambda(Ab,Ag)=\sum_{i=1}^{L}\delta_i,\ where \begin{cases}1 & : \ if\ \ Ab_i\neq Ag_i\\ 0 & : \ otherwise\end{cases}$$

where Eq. 2 is the Euclidean distance, Eq. 3 the Manhattan distance and Eq. 4 the hamming distance. In these equations, L is the length of the binary string feature vector and Ab_i and Ag_i refer to the *ith* feature of those vectors. The Hamming distance can be computed by applying the exclusive-or operator (XOR) to the binary strings. For example, the test vectors *v* and *w* and their computation of hamming distance, λ(v, x), is illustrated in figure 3.

As mentioned earlier, the affinity between antibodies indicates the similarity of antibodies, the greater the affinity value, the greater similarity between antibodies. The affinity between two antibodies *v* and *w* noted by $B_{v,w}$ is defined the reciprocal of the hamming distance as Eq. 5:

$$B_{v,w}=\frac{1}{\lambda(v,w)} \tag{5}$$

The mean distance between two vector *v, w* with the length of L is defined by Eq. 6.

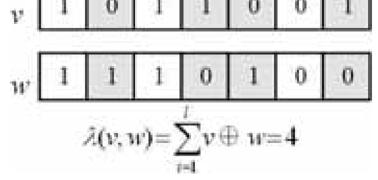

Fig. 3 Hamming distance between two antigens

$$\gamma_{v,w} = \frac{\lambda(v,w)}{L} \tag{6}$$

The density of antibody *v*, called C_v, can be defined as Eq. 7 (Wei, *et al.* 2003):

$$C_v = \frac{\sum_{w=1}^{N} \gamma_{v,w}}{N} \tag{7}$$

where N is the number of antibodies. The higher density of one antibody (C_v), the smaller similarity between this antibody and others. In the others words, the antibodies with the low density have more similar to other antibodies; therefore, to keep the diversity of antibodies population, the antibodies with low density are eliminated in the next GA generation. The new random antibodies are substituted for the eliminated antibodies. Suppressing the low density antibodies can greatly keep the diversity of population and avoid trapping into the local optimal solutions (Wei, *et al.* 2003).

Calculation of Selection probability:

The GA uses selection operator to simulate natural evolution. In GA, individual with high fitness is inherited to the next generation with greater probability. Usually, chromosomes with high fitness are selected for crossover to converge faster to best solution. Chromosomes with high fitness should not be selected for mutation to prevent the danger of diverting from good solutions in the search space. Therefore, chromosomes with low fitness are usually selected for mutation. All selection methods are based on the fitness of chromosomes. The disadvantage of selecting chromosomes with high fitness is the probability of less diversity in the search space. If the selection operator of GA is used, the result probably trip into local optimum trap. The selection function which is applied in this paper is similar to the one in (Jiaxin, *et al.* 2006):

$$p(x_i) = \alpha \times \frac{F(x_i)}{\sum_{j=1}^{N} F(x_j)} + \beta(1 + C(x_i))$$

$$\alpha + \beta = 1 \tag{8}$$

The optimum range of α and β is (0.35~0.65). In this paper, we consider α = 0.6. The selection probability of antibody is composed of its fitness probability and density probability. Eq. (8) shows that the bigger the fitness of an antibody is, the bigger its selection probability and the higher of an antibody density, the higher its selection probability. As a result, not only antibodies with high fitness are saved, but also the diversity of antibody is guaranteed by the promotion and suppression between antibodies based on their mean distance. *Promotion* of antibody means the increase of its selection probability and *suppression* means the decrease of its selection probability.

Crossover and Mutation:

Crossover is the key to genetic algorithms power that is to exchange corresponding genetic properties from two parents, to allow useful genes on different parents to combine in their offspring. Most common crossover types are one-point, two-point and uniform crossover. In this paper, as shown in Fig. 4, two-point crossover is used as a crossover operator.

The next genetic operator to be applied to the population is mutation, which employed to introduce diversity into the population. As shown in Fig. 5, in the random mutation, random genes in the chromosome are selected and each gene's value is replaced with its complement or a new random value (with probability p_m). The role of mutation in GA is to prevent the premature convergence of GA to suboptimal solution by restoring lost or unexplored genetic material into the population. The mutation probability (p_m) is proportional to the affinity between the antibody and antigen. Small value of p_m is essential for the successful working of GA, because the moderately large value of p_m promotes the expensive recombination of schemata, while small value of p_m is necessary to prevent the disruption of schemata. Especially, p_m must decrease to survive the individuals with high fitness; therefore p_m needs to be zero for the solution with the maximum fitness. The expression for p_m is given in (Li, 2005) as shown in Eq. 10:

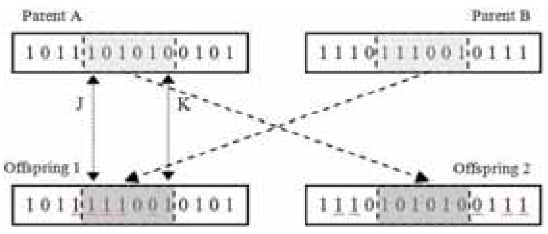

Fig. 4: Two-point crossover

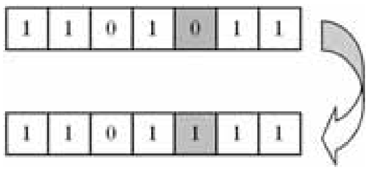

Fig. 5: Mutation in binary string

$$p_m = \begin{cases} \alpha(0.5 - f_d), & 0 \le f_d \le 0.5 \\ \alpha(1 - f_d)^2, & 0.5 \le f_d \le 1 \end{cases}$$
$$0 < \alpha < 1$$

where f_d is the normalized fitness value and α is a number between 0 to 1. In this paper Gaussian mutation is taken as the mutation operator.

RESULTS

The proposed algorithm is simulated using a computer code with C# tools. The experiments are carried out on ISCAS'85 combinational benchmark circuits. The computer used to run the simulation has an AMD processor with the clock rate of 1.8 GHz and a 512 Mbyte memory. For fault simulation and fitness evaluation, we used executable file of FSIM parallel fault simulator. In the first experiment, the minimum number of test vectors is determined to detect all detectable faults. The number of detected faults and the number of test vectors generated are shown in table 2. The population size are used is the same as Table 1.

Table 2: Simulation Results on ISCAS85 Benchmarks

Circuit	Detected Faults	Test size (Saab, D. G., *et al.,*1992)	Detected Faults	Tests size (Pomeranz and S.P. Reddy, 1997)	Tests size (Ivask, E., *et al.,* 1998)	Tests size Proposed Algorithm	Reduction in Test size compared With ((Ivask, E., *et al.,* 1998) (%)
C432	519	72	520	76	46	43	6.5
C499	749	553	750	67	54	48	11.1
C880	937	229	942	115	85	53	37.6
C1908	1852	253	1870	176	126	120	4.8
C3540	3277	452	3291	276	149	134	10.1
C5315	5258	682	5291	221	120	96	20
C6288	7709	131	7709	-	23	18	22

Table 3: Reductions using proposed method in comparison with)12(for 90% fault coverage

Circuit	Test size (Rad, M.A, and S.M Eshgh, 2007)	Test size Proposed Algorithm	Test size Improvement (%)	Execution Time (min) (Rad, M.A, and S.M Eshgh, 2007)	Execution Time (min) Proposed Algorithm
C17	4	3	25	1.16	0.18
C432	27	20	26	6.78	1.08
C499	20	15	25	4.28	0.86
C880	25	16	36	8.85	0.96
C1355	33	26	21	7.21	1.9
C3540	80	57	40	13.55	8.8

The test sets are significantly smaller than those reported for CRIS (Saab, *et al*. 1992) and in addition, the fault coverage obtained is higher for all circuits. However, our test generation time is longer than those of CRIS (Saab, *et al*. 1992). That is for two reason; first, the authors in (Saab, *et al*. 1992) use a SUN SPARC SLC workstation, where we used a PC with the about species. Second, we use the executable file of FSIM simulator instead of its source code. We compared our experimental results to results reported in (Pomeranz and Reddy, 1997), where the key feature is keeping certain inputs together (in order to better propagate fault effect) during reproduction process. As shown is table 2, the approach given here uses up to 2 times less of test vectors for all circuits. The improvement of our method in test size, compared to (Ivask, *et al*. 1998) is up to 16% in average. In another comparison, we compared our experimental result with results reported in (Rad and Eshgh, 2007), which fault coverage has been set to 90% and used FSIM fault simulator with an Intel Celeron processor with the clock rate of 1.5 GHz and a 512 Mbyte memory. As shown in table 3, our immune genetic based algorithm has the average reduction of 28% in test size and up to 4 times less in test time, compared to (Rad and Eshgh, 2007).

DISCUSSIONS

In past decades, extensive attempts have been made to find practical solutions for circuits test generation. ATPG tools can reduce the amount of effort and cost of test generation. Random test generation is a simple process where vectors are generated without targeting any specific faults. Genetic algorithm is a random search which is specially suited to solve large scale combination optimization problem. Since test generation is a search process over a large vector space, it is an ideal candidate for GAs. Using genetic algorithm in test pattern generation is effective in achieving a compact test set with high fault coverage and reducing the generation time. In this paper, a new immune genetic based algorithm for automatic test pattern generation (ATPG) is introduced. Our experimental results showed that a new immune genetic algorithm-based method is more efficient in test generation in comparing with non-immune algorithms. Our experimental results show an average 25% reduction in test size and up to 4 times less in test time in comparison with results reported in (Rad and Eshgh, 2007).

REFERENCES

Breuer, A. and A.D. Friedman, 1997. Diagnosis and reliable design of digital systems: Computer Science Press Inc.

Arslan, T. and M.J., O'Dare, 1997. A genetic algorithm for multiple fault model test generation. In Proc. of Second International Conference on Genetic Algorithms in Engineering Systems. Innovations and Applications, pp: 462-466.

Azimipour, M. and M. Eshghi, 2008. Parallel Circular-Scan Architecture. Asian Network for Scientific Information (ANSI), Journal of applied sciences, pp: 1-8.

Azimipour, M., M. Eshghi and A. Khademzadeh, 2007. A Modification to Circular-Scan Architecture to improve test data compression. IEEE CS Proc. Of 15th Int. conf. on advance computing and communication, pp: 27-33.

Al-Yamani, A., Erik Chmelar and Mikhail Grinchuk, 2005. Segmented Addressable Scan Architecture. In Proc. of VLSI Test Symposium, pp: 405-411.

Goel, P., 1981. An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits. IEEE Trans. On Computers, vol. C-30, pp: 215-222.

Abramovici, M., M.A. Breuer and A.D. Friedman, 1994. Digital Systems Testing and Testable Design. Wiley-IEEE Press.

Seshu, S. and D.N. Freeman, 1962. The diagnosis of asynchronous sequential switching systems. IRE Trans. On Electronic Computing, Vol. EC-11, pp: 459-465.

O'Dare, M.J. and T. Arslan, 1994. Generating test patterns for VLSI circuits using a genetic algorithm. IEEE Electronic Letters, 30(10): 778-779.

Ivask, E., J. Raik and R. Ubar, 1998. Comparison of Genetic and random Techniques for Test Pattern generation. Proceeding of the 6th Baltic electronics conference, pp: 163-166.

Rudnick, E.M., J.H. Patel, G.S. Greenstein and T.M. Niermann, 1997. A Genetic Algorithm Framework for Test Generation. IEEE Transactions on computer-aided design of integrated circuits and systems, 16(9): 1034-1044.

Rad, M.A. and S.M. Eshgh, 2007. A heuristic algorithm with a revocation based GA for test pattern generation of VLSI circuits. IEEE Proc. Of Int. Conf. On Electrical Engineering, pp: 1-5.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Reading, MA: Addison-Wesley.

Li, Y., Y. Dai and X. Ma, 2005. A heuristic immune-genetic algorithm for multimodal function optimization. In Proceeding of CIMCA/IAWTIC conference, pp: 36-40.

Jiao, L. and L. Wang, 2000. A novel genetic algorithm based on immunity. PERLINK"http://www.informatik.uni-trier.de/~ley/db/journals/tsmc/tsmca30.html"\l"JiaoW00"IEEE Transactions on Systems, Man and Cybernetics, Part A 30(5): 552-561.

Breuer, M.A., 1971. A random and an algorithmic technique for fault detection test generation for sequential circuits. IEEE Trans. on Computer., vol. C-20, pp: 1364-1370.

Schnurmann, H.D., E. Lindbloom and R.G. Carpenter, 1975. The weighted random test-pattern generator. IEEE Trans. Comput., vol. C-24, pp: 695-700.

Snethen, T.J., 1977. simulator-oriented fault test generator. in Proc. Of Design Automation Conf., pp: 88-93.

Agrawal, V.D., K.T. Cheng and P. Agrawal, 1989. A directed search method for test generation using a concurrent simulator. IEEE Trans. Computcr-Aided Design, 8(2): 131-138.

Simões, A.B. and E. Costa, An Immune System-Based Genetic Algorithm to Deal with Dynamic Environments: Diversity and Memory, Proceedings of the Sixth international conference on neural networks and genetic algorithms (ICANNGA03), Springer, pp: 168-174.

Chattopadhyay, S. and N. Choudhary, 2003. Genetic algorithm based approach for low power combinational circuit testing. In proceeding of the IEEE 16th Int. Conference on VLSI Design, pp: 552-557.

Castro, de, L.N. Timmis and Jon Timmis, 2003. Artificial immune systems as a novel soft computing paradigm. Springer soft computing Journal, 7(8): 526-544.

Wei, H., Xu Chunli, Zhang Jianhua and Hu Shan'ang, 2003. Study of reactive power optimization based on immune genetic algorithm. IEEE PES transmission and distribution conference and exhibition, pp: 186-190.

Jiaxin, Y., C. Baichao, T. Cuihua and S. Yu, 2006. The research of inverter's control based on immune genetic algorithm. In Proc. Of IEEE Conference on Industrial Electronics and Applications, pp: 1-6.

Saab, D.G., Y.G. Saab and J.A. Abraham, 1992. CRIS: A test cultivation program for sequential VLSI circuits. in Proc. Int. Conf. Computer-Aided Design, pp: 216–219.

Pomeranz and S.P. Reddy, 1997. On Improving Genetic Optimization based Test Generation. Proc. of the IEEE European Design and Test Conf., pp: 506-511.