IJCTA, 9(12), 2016, pp. 5591-5598

Test Vector Generation Using Genetic Algorithm for Fault Tolerant Systems

***K. Jamal ** Dr. P.Srihari ***G Kanakasri**

Abstract : In this paper, a Genetic Algorithm (GA) is proposed in order to generate test patterns automatically for finding the faults present in memory. Genetic algorithm solves effectively many search and optimization problems for fault tolerant systems. For many complex problems, genetic algorithm can find an optimal solution. By deriving minimal test sets, we can reduce the post production cost for testing memory. The GA proves to be an efficacious algorithm in detecting perfect number of test patterns from the highly complex problem space. As the genetic algorithm promotes, a well organized set of test patterns are produced, by observing the solution space for test patterns that detect the more number of remaining faults present in the fault list . Experimental results on plain memory proves that the Genetic algorithm diminish the complexity of the circuits and also the execution time. The GA is realized using System Verilog. Moreover, the test sets generated by the proposed algorithm are more tightly packed.

Keywords : GA, Test pattern generation, Evolutionary Algorithm, stuck-at-faults.

1. INTRODUCTION

During 1980s and 1990s, the practice and theory of testing electronic devices have changed a lot. As technology advances, the complexity of the devices increases and so we have the testing problems. This is because with the increasing level of integration, the ratio of I/O pins to the number of internal nodes decreases and it leads to a decreased observability and controllability of internal nodes. In other words, if internal structure of network is more complex then observing the behavior of internal node is more complicated.

Genuine and faultless operations of integrated circuits are of very important concern as they are used in flight control systems, medical, military, manufacturing robots [2], carpool service[3]. And also genetic algorithms have been applied to other problems such as WSNs in general localization [4], positioning and node deployment [5][6], and TDMA scheduling [7] Before using integrated circuits in a fully operational environment, we have to test their performance in order to find any faults.

Nowadays testing of integrated circuits is an important problem because it takes nearly a remarkable percent of the total production and design costs of an ASIC. Moreover, several independent studies say that the cost of detecting a fault in a circuit is very much cheaper than to detect the same fault inside same circuit as a subsystem contained in a system.

A VLSI circuit can be tested by applying a sequence of values to its input pins and observe the values at the output pins of the circuit. Then we get the values observed on the Output Pins of the fault-free circuit are different from the values appearing on the same Pins of any faulty circuit.

Apart from the high fault coverage rate, an efficient ATPG tool diminishes the test pattern generation cost and time. Deterministic approach, simulators etc are the different approaches of ATPG. The objective of the proposed method should be both to improve fault coverage and to reduce execution time.

It is extremely difficult for the identification of general problem. An NP-complete problem is defined as improving a diagnostic sequence of tests and a long standing problem [8] is defined as obtaining a method to

* Department of ECE GRIET, Hyderabad, India kjamal24@gmail.com

** Department of ECE GCET, Hyderabad, India mail2pshari@yahoo.com

*** M.Tech-Scholar GRIET, Hyderabad, India kanakasri.gorijavolu@gmail.com

improve practical diagnosis. Today, people are keen towards Artificial intelligence methods and one among them is evolutionary algorithms or frequently called as genetic algorithms.

Based on the Darwinian theory of evolution and natural selection [9], genetic algorithms are proved that they are practical, robust optimization tools and are well suited for deriving optimal test sets. In [1], an accelerated GA has been iteratively applied to reduce rough sets attributes.

2. GENETIC ALGORITHM

John Holland, his colleagues and his students at the University of Michigan had developed the Genetic algorithms. There are two objectives of their research: (1) the flexible nature of natural systems can be carefully explained and summarized, and (2) artificial systems software can be designed such that they can preserve the essential mechanics of natural systems.

The principle of development inspires stochastic optimization technique which is nothing but a genetic algorithm. Genetic algorithm is one of the most popular methods among all evolutionary computations. The representation used in the traditional genetic algorithm is a fixed-length bit string. Each bit in the string is supposed to represent a distinct characteristic of an individual, and the value preserved in each position indicates how that characteristic is expressed in the solution. Generally, the string is estimated as a group of structural characteristics of a solution that have minute or no interactions.Holland expressed that under certain modifications, by controlling the structures properly, we can achieve rapid improvements of bit strings. Generations of bit strings "advances" as populations of animals do. An essential, traditional outcome highlighted by Holland was that in large and complicated search spaces the genetic algorithms will converges to exact or near optimal solutions.

A Genetic algorithm must have following components in order to solve a problem:

(*a*) Representation. (*b*) Initialization.

(*c*) Fitness function. (*d*) Reproduction

A. Representation

In genetic structure, one possible solution to the problem is called individual or also referred as chromosome. For a single problem we have different solutions and one solution is more optimal than other. All chromosomes together form a population. We represent the individuals in binary format that is bit strings of 1's and 0's. we can code variety of information by using bit strings and bit strings have been shown to be powerful representations in unexpected domains. A good thing is known about genetic operators and parameter values are that it works well with the properties of bit string representations of genetic algorithms [10].

B. Initialization

Initializing the population randomly is suitable for research purpose. It is a good test of algorithm for Moving from a randomly created population to a well adapted population. The search and recombination mechanism of genetic algorithm produces the critical characteristics of final solutions. Generally it is good to use direct methods for initializations in order to maximize the quality and speed of the final solution that is we can use outputs of another algorithm for initialization.

C. Fitness function

Fitness function is used to judge the fitness of the chromosomes in a population. Better solutions will get higher score. Evaluation function controls population towards progress because good solutions will be chosen during selection process and poor solutions will be discarded.

D. Reproduction

During reproduction period of genetic algorithm, chromosomes are selected from population and recombined, and then we obtain children which will be added to the next generation. We can select the parents which can have more fitness value randomly. Good chromosomes will probably be selected more number of times for reproduction and poor ones may not be selected even a single time. We will select two chromosomes as parents and then their genes are exchanged using the operations of crossover and mutation.

1. Selection

Selection process finds two or more chromosomes for crossover operation. The selection pressure is the level to which the better chromosomes are recommended that is the higher the selection pressure then more the better individuals are preferred. This selection pressure manages the genetic algorithm to improve the population fitness over subsequent generations. The selection pressure largely determines the convergence rate of genetic algorithm. Higher convergence rates are due to higher selection pressure. Under a broad range of selection pressure [11] Genetic algorithms are able to identify favorable solutions. The genetic algorithm will take longer time in order to find the optimal solution if the selection rate is too low.

Various possible selection strategies are:

(*a*) Roulette wheel selection : It is a traditional gambler's roulette wheel that is a rotating circle is divided into several sections of equal size with numbering. The person in charge of a game sets the wheel rotating and throws t h e marble onto wheel. Marble comes to rest in one of the numbered sections when the motion of the wheel stops. Roulette-wheel selection is the effortless selection and it is also called stochastic sampling with replacement.

In case of genetic algorithms, chromosomes are selected using roulette wheel for further reproduction. In roulette wheel selection process, each chromosome is mapped to adjacent segments of a wheel, such that each chromosome segment size is equal to its fitness value. Then we throw a marble on the rotating wheel and af ter some time the wheel stops rotating then the marble came to rest on any one of the segments of the wheel. Then the chromosome corresponding to that segment is taken as the parent for the next generation and this operation is repeated till the required number of individuals is received. The chromosomes that has highest fitness will have more chance for reproduction and ones that have least fitness will have less chance for reproduction.

(*b*) Stochastic universal selection : In Stochastic universal selection the individuals are mapped to adjacent segments of a wheel, such that each individual's segment size is equal to its fitness absolutely same as in roulette-wheel selection. Here how many individuals we require for reproduction those many pointers are placed around the wheel. But the pointers are placed in such way that there exists equal space between them. By rotating the wheel single time we form the population for reproduction

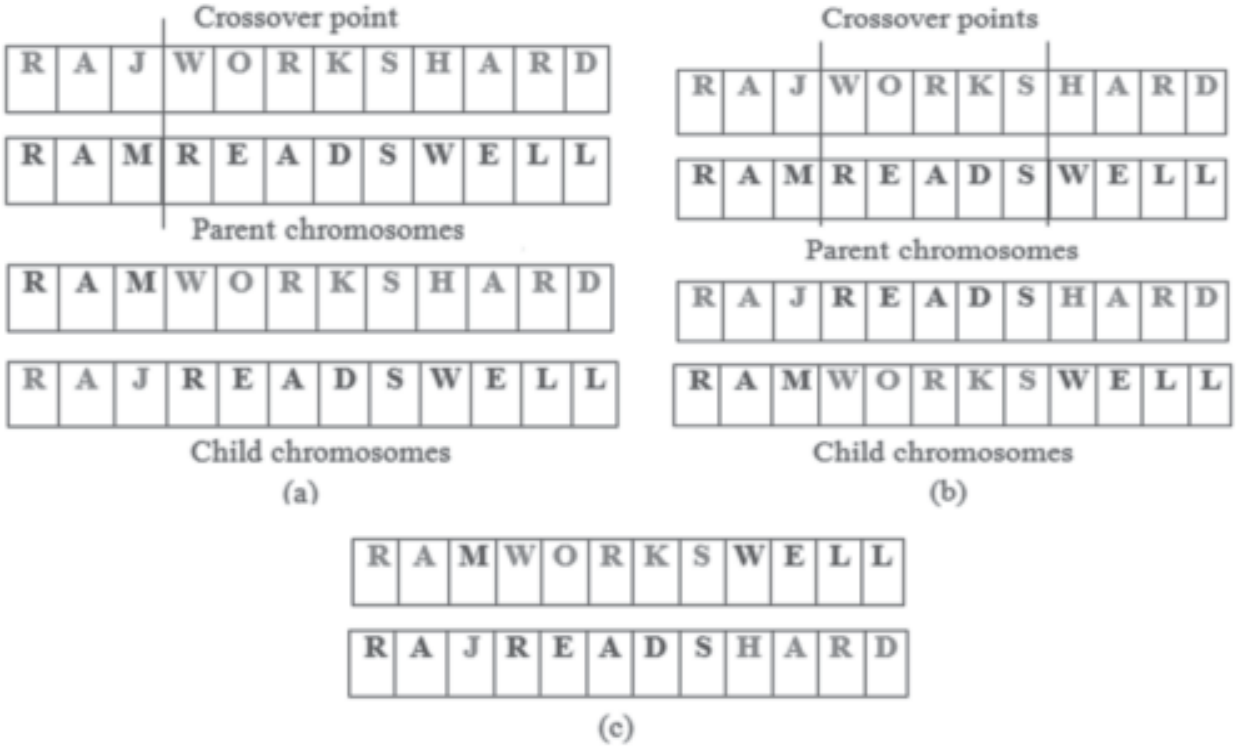

Fig. 1. (*a*) One point crossover (*b*) Two point crossover (*c*) Uniform crossover

(*c*) Tournament selection : For conducting tournament,'*s*' individuals are taken at random, and the effective one is selected as the winner of that tournament. By changing tournament size we can adjust its selection intensity.

Among all '*s*' tournament competitors the champion of the tournament is the individual that has highest fitness value. And the winner of each tournament is inserted into mating pool. Now the mating pool is filled with all tournaments winners and it has higher average fitness compared to average population fitness. The difference between average fitness of population and average fitness of mating pool provides selection intensity. Selection intensity steers the genetic algorithm to enhance the fitness of each subsequent generation. By decreasing (increasing) the tournament size'*s*', we can decrease (increase) tournament selection intensity. Here the winner of a larger tournament will have a higher fitness than the winner of a smaller tournament.

2. Crossover

Crossover is nothing but an offspring inherits the some properties from one parent and other properties from another parent that is exchanging of genetic material in two parents. Different strategies are there for selecting two parents as parents for the next generation.

Genetic algorithms have power due to crossover operation. More frequently successful parents reproduce. Genetic algorithm combines useful properties of two parents. Typical types of crossovers are shown in figure 1. Each character in the string represents the genetic material of that string. Here, we can see that how the genes are mixed in every particular case of crossover.

3. Mutation

When crossover operation is performed on same set individuals then in the end we get children are same as parents to avoid this we perform mutation after crossover operation. And one more thing is during crossover operation we lost some of the features of parents, in order to retain those features we perform mutation operation. In case of binary strings Mutation is nothing but inverting a bit in the chromosome as shown in figure 2 and the complete GA cycle is shown in figure 3.

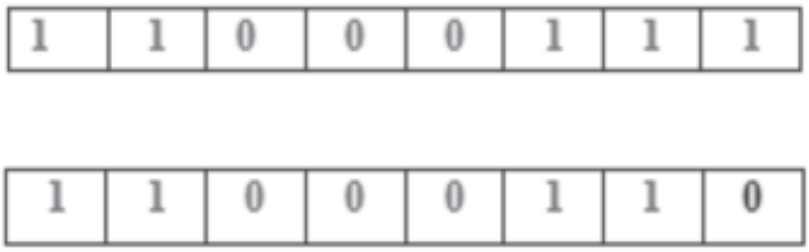

Fig. 2. Mutation operation.

First read the number of generations which is a condition for terminating the process and initially set '*n*' to 0. Where *n* represents count value which represents number of generations completed. Initially some chromosomes (let say 5) were generated as parents by using random generation method. Once parents are ready then they can produce the children *i.e.*, the next generation. So, '*n*' value is incremented. Randomly select two chromosomes from parents and apply two points crossover as shown in figure 1(*b*) and get the two Offspring's. If two Offspring's are same then apply mutation operation as shown in figure 2 by randomly selecting the bit position. Out of two Offspring's, randomly select one and discarded the other. This step is repeated until we get the required number of Offspring's (let say 15). Once all chromosomes are obtained then calculate the fitness value for each and every chromosome *i.e.*, nothing but transition count. After calculating fitness values for all chromosomes, arrange them in ascending order based on fitness value of each chromosome.

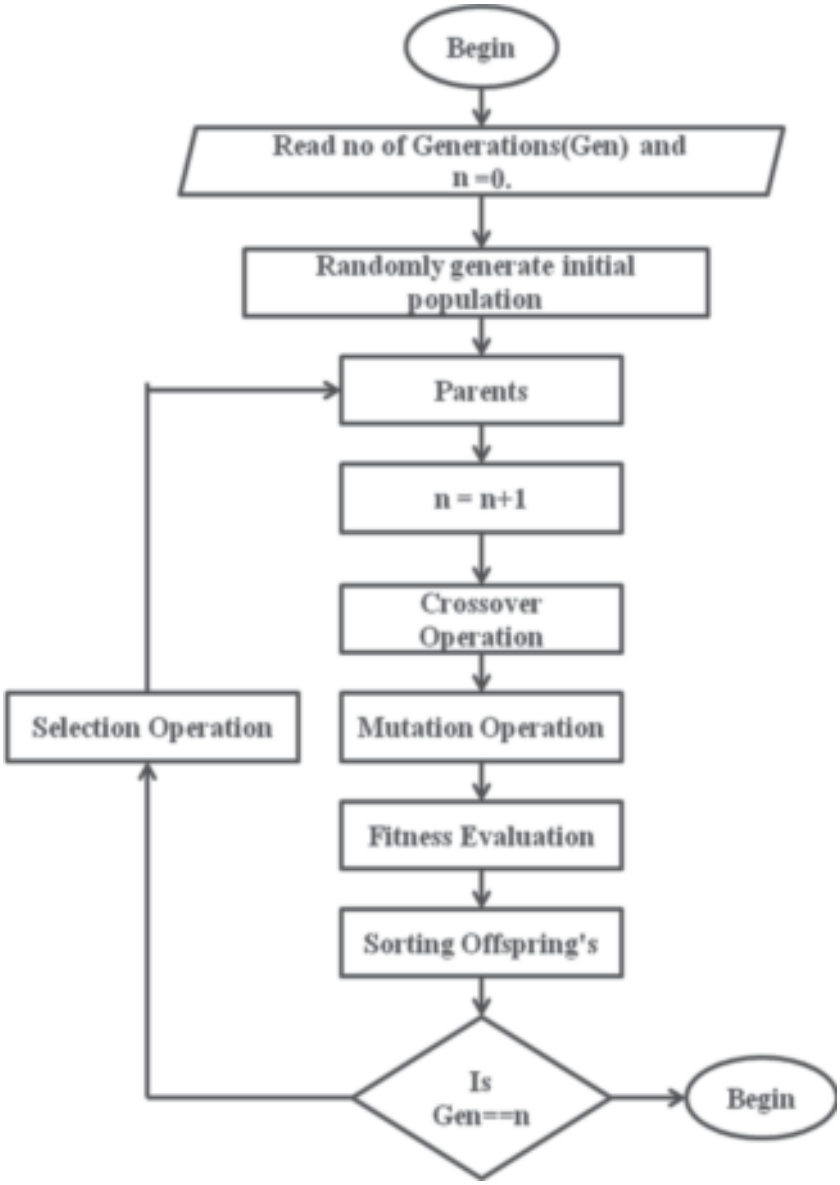

Fig. 3. Genetic algorithm cycle.

After arranging all the chromosomes in order check the condition *i.e.*, number of generations are equal to or not equal to *n*. If yes terminate the process else select the top (let say 5) chromosomes as parents for next generation and go to step 3 by discarding the rest of chromosomes.

3. EFFECTS OF GA OPERATORS

Now first see the individual effects of selection, crossover and mutation on schemas existing in population and after that the combined effect of crossover, selection and mutation on the schemas present in population will be seen.

Suppose at any given time '*t*' there are '*x*' patterns of schema S in the population. The nit can be represented as x = x (S,*t*). At different time moments we have different number of schemas. In selection process based on fitness function, a string is copied into next generation. Probability that the string Bi selected is

$$p_i = \frac{f_i}{\sum f_i}$$

Where f_i represents the fitness of the individual string and $\sum f_i$ represents the sum of the fitness of individual strings present in the population.

If the size of the population is 'n' then we expect to have exemplars of schema S present in a population at time moment '$t + 1$',

$$x(S, t + 1) = x(S, t)^* n \frac{f(S)}{\sum f_i} \quad ...(1)$$

where $f(S)$ represents the average fitness of the string denoted by schema S in a time moment 't'.

The average fitness of the population can be given as follows :

$$f = \frac{\sum f_i}{n}$$

Then we can rewrite the expression (*i*) as follows :

$$x(S, t + 1) = x(S, t) \frac{f(S)}{f} \quad ...(2)$$

Above expression (2) shows how selection process effects schema that is the schema which have fitness value greater than the population's average, will get more number copies in next generation. Similarly the vice versa.

If we execute only selection operation on a set of chromosomes then we get same population in next generations. Then there is no improvement in genetic algorithm. This can be avoided by using crossover operation. Now we are going to see the affects of crossover on each chromosome.

If we perform crossover operation on chromosome S such that the crossover point falls in its useful length then the probability that the chromosome S can be destructed is given by the following expression:

$$p_d = \frac{\delta(S)}{L - 1}$$

If the crossover point falls outside the useful length of chromosome S then the probability that the chromosome S is survived is given as follows:

$$p_s = 1 - \frac{\delta(S)}{L - 1} \quad ...(3)$$

Let us assume that pc be the probability that crossover takes place then for any type pairing, that is crossover point lies inside or outside its useful length, the probability that the schema S survive is given below:

$$p_s \geq 1 - p_c^* \frac{\delta(H)}{L - 1} \quad ...(4)$$

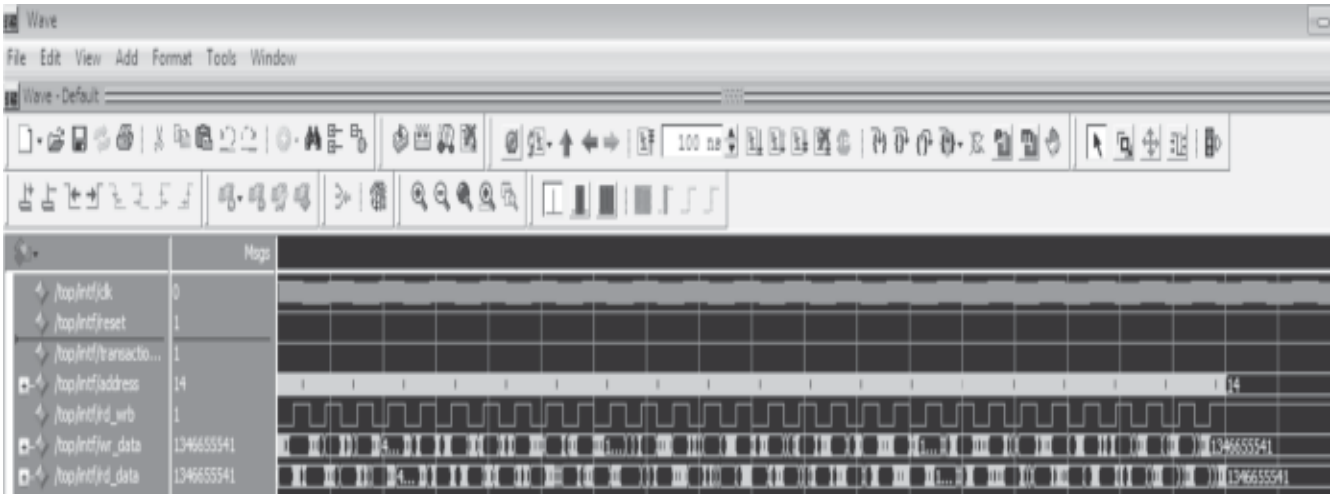

Fig. 4. Simulation results of memory with-out faults

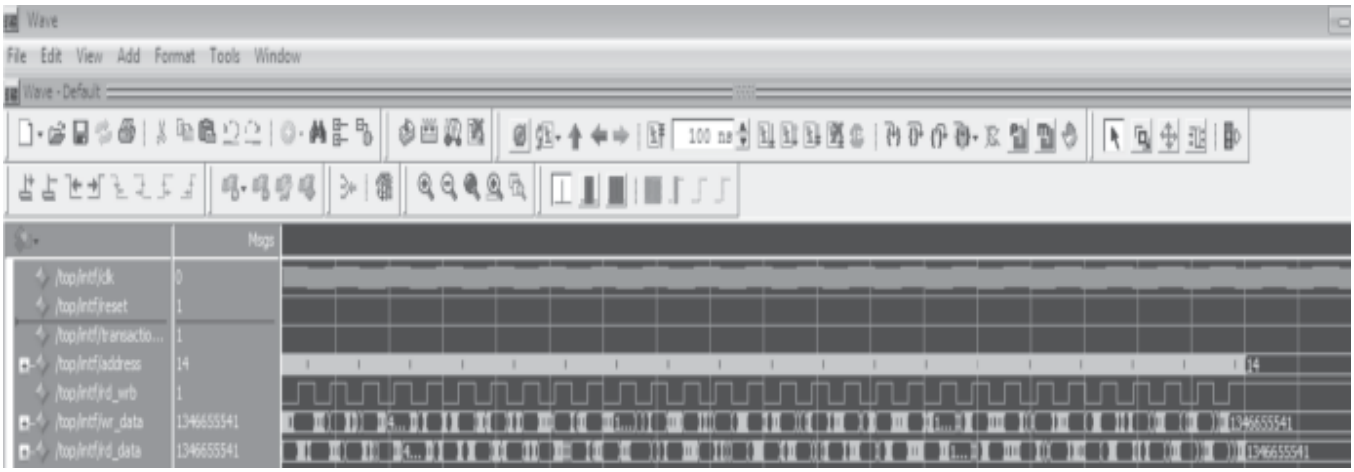

Fig. 5. Simulation results of memory with faults

Suppose the selection and crossover operations are independent to each other then combined effect of selection and crossover is given by the following evaluation:

$$n(\mathrm{S}, t+1) \geq n(\mathrm{S}, t)^* \frac{f(s)}{f}\left[1 - p_c \frac{\delta(\mathrm{S})}{\mathrm{L}-1}\right] \quad ...(5)$$

Mutation operation alters one character present in a string randomly with probability pm. The probability that single instantiated position in a string unaltered is $(1 - pm)$. Here the schema S is survived only when all its instantiated positions are unaltered. Then the probability that the schema S is survived is obtained by multiplying surviving probability $(1 - p_m)$ of single instantiated position with o(S) times. We obtain surviving probability as :

$$(1 - p_m)^{o(S)}$$

Since pm is very small, we can approximate the above equation as (1–o(S)*pm).

T he combined effect of selection, cross over and mutation operations on schema S is given by the following equation:

$$n(\mathrm{S}, t+1) \geq n(\mathrm{S}, \mathrm{t})^* \frac{f(\mathrm{S})}{f}\left[1 - p_c \frac{\delta(\mathrm{S})}{\mathrm{L}-1} - \mathrm{o}(\mathrm{S}) p_m\right] \quad ...(5)$$

4. RESULTS

Figure 4 shows that the memory is fault free. That is at memory location (address) 14 and rd_wrb is 1 then what ever the data written at this location is read from that position. This shows that memory location 14 is fault free. Similarly remaining all memory locations are verified. For a fault free memory as shown in above figure whatever the data we wrote into each and every memory location that data is read back from memory. The above figure 5 shows the simulation results of a memory having faults. We place a stuck at fault in memory location 1. In memory location 1 at 8120ns time we wrote 1413829941 but while we read the data from that location we get 134665541 instead of 1413829941 which indicates an error present in the memory. Similarly at 8269 ns time we wrote 1346655541 but while we read the data from that location we get 1413329943 instead of 134665541.

5. CONCLUSION

An efficient Genetic Algorithm for memory testing with high speed is proposed here. By using genetic algorithm we get near optimal or optimal solutions to a given problem with less span of time.Genetic algorithms are best for searching new results and causing use of results that have worked well in the past that is it takes decisions based on previous solutions. Initially Genetic algorithm may not find the solutions to a problem but as the number of generations increases it can find the solutions.

6. REFERENCES

1. Abdel-Rahman Hedar, Mohamed Adel Omar, Adel A. Sewisy. Rough Sets Attribute Reduction Using an Accelerated Genetic Algorithm. 2015 IEEE SNPD 2015, June 1-3 2015.
2. Chiu-Hung Chen, Tung-Kuan Liu, and Jyh-Horng Chou. A Novel Crowding Genetic Algorithm and Its Applications to Manufacturing Robots. IEEE Transactions On Industrial Informatics, Vol. 10, No. 3, August 2014.
3. Shih-Chia Huang, Ming-Kai Jiau, and Chih-Hsiang Lin. Optimization of the Carpool Service Problem via a Fuzzy Controlled Genetic Algorithm. IEEE Transactions on fuzzy systems, 2014.
4. S. H. Chagas, J. B. Martins, L. L. de Oliveira, "An approach to localization scheme of wireless sensor networks based on artificial neural networks and genetic algorithms," in Proc. 2012 IEEE 10th International New Circuits and Systems Conference, 2012.
5. P. Sun, J. Ma, and K Ni, "A genetic simulated annealing hybrid algorithm for relay nodes deployment optimization in industrial wireless sensor networks," in Proc. 2012 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, 2012.
6. B. Jiang, W. Zhang, and P. Zhang, "Research on an improved genetic algorithm which can improve the node positioning optimized solution of wireless sensor networks," in Proc. International Conference on Computational Science and Engineering, 2009, 2009.
7. M. Ziari, M.-H. Yaghmaee, Z. Davarzani, and M.-R. Akbarzadeh-T, "TDMA scheduling in wireless sensor networks using hybrid of genetic algorithm and particle swarm optimization," in Proc. 2010 6th International Conference on Networked Computing, 2010.
8. W. R. Simpson, J. W. Sheppard "System Test and Diagnosis".
9. Pomeranz, S. M. Reddy, "On improving Genetic Optimization based Test Generation", Proc. Of European Design and Test Conference 1997, pp. 506-511.
10. J.H. Holland, "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975 .
11. E. M. Rudnick, J. H. Patel, G. S. Greenstein, T. M. Niermann "Sequential Circuit Generation in a Genetic Framework", 31st ACM/IEEE Design Automation Conference.