International Journal of Computer Applications (0975 – 8887)
Volume 12– No.10, January 2011

A Genetic Algorithm based Two Phase Fault Simulator for Sequential Circuit

Dhiraj Sangwan
Assistant Professor
ECE Department
MITS, Laxmangarh

Seema Verma
Associate Professor
Department of Electronics
Banasthali Vidaypeeth

Rajesh Kumar
Associate Professor
Department of EE
MNIT, Jaipur

ABSTRACT

Testing has become an important design step now days in digital circuit. A gate level fault simulation environment based on realistic fault models has been presented in this paper. A Genetic Algorithm (GA) is proposed which allows having fault simulation with conditional execution of test vector under 2 phase scheme. By using this approach a random search of test vectors is possible without being caught in a local minima or maxima. The award of fitness to the vector set allows having a selection of test vectors with high fault coverage and with large fault detection scores. Experimental results are provided which shows that the proposed technique can be employed for the detection of faults in a sequential circuit with high fault coverage.

General Terms

Sequential Circuit, Flip- flop, Algorithms et. al.

Keywords

Genetic Algorithm, Sequential Circuits, Automatic Test Pattern Generator, Fault coverage, Circuit Under Test, Flip-flop

1. INTRODUCTION

Digital systems produced today are extremely complex in nature. With increase in complexity & density of VLSI networks in almost all Electronics appliances, a very high demand is put on the reliability of such networks. A fault simulator is a vital part of Automatic Test Pattern Generator system (ATPG) system [1, 2]. Various faults might occur while laying out such VLSI circuits. Common faults found are the stuck-at fault and gate delay faults [3].

Stuck-at faults are electrical failures due to physical failure mechanisms. Stuck at faults can be stuck-on(stuck-at-1)or stuck-open faults(stuck-at-0).Unless there are logical redundant connections, any stuck-at fault causes electrical failure of the whole network. Hence physical testability of VLSI circuits for stuck-at faults is very important [4].

Detection of such faults requires an optimum number of test vectors to be selected from all possible combinations. A GA based sequential circuit fault simulator is employed to evaluate fitness of each candidate vector and select best vector to apply in each time frame [5, 6]. We experimented with various GA parameters namely population size, number of generations, probability of crossover and mutation, fitness award value, different crossover schemes and adaptive and non-adaptive crossover and mutation strategy. The good fault coverage is achieved in both test circuits namely 4 bit parallel load shift register and vending machine controller with relatively less execution times. The implementation of the proposed algorithm is done using MATLAB.

1.1 Genetic Algorithm Basics

GA uses a method of inheritance to solve various problems, such as those involving adaptive systems. Of course, since GAs are heuristic procedures, they are not guaranteed to find the optimum, but experience has shown that they are able to find very good solutions for a wide range of problems. GA work by evolving a population of individuals over a number of generations. Since selection is biased toward more highly fit individuals, the average fitness of the population tends to improve from one generation to the next. The fitness of the best individual is also expected to improve over time, and the best individual may be chosen as a solution after several generations [7].

The initial population is typically generated randomly, but it may also be supplied by the user. A highly fit population is evolved through several generations by selecting two individuals, crossing the two individuals to generate two new individuals, and mutating characters in the new individuals with a given mutation probability. Selection is done probabilistically but is biased toward more highly fit individuals, and the population is essentially maintained as an unordered set. Distinct generations are evolved, and the process of selection, crossover, and mutation are repeated until all entries in a new generation are filled. Then the old generation may be discarded. New generations are evolved until some stopping criterion is met. The GA may be limited to a fixed number of generations, or it may be terminated when all individuals in the population converge to the same string or no improvements in fitness values are found after a given number of generations [8]. Since selection is biased toward more highly fit individuals, the fitness of the overall population is expected to increase in successive generations. However, the best individual may appear in any generation. GA uses probabilistic rules and an initial random population. Thus, early on, the search may proceed in any direction and no major decision is made in the beginning. Later on, when population has converged in some locations the search direction narrows and a near-optimal solution is found. This nature of narrowing the search space as generation progresses is adaptive and is a unique characteristic of GAs [8].

This generational process is repeated until a termination condition has been reached. If the termination criterion is not satisfied, the population of solutions is modified by three main operators and a new (and hopefully better) population is created. The generation counter is incremented to indicate that one generation of GA is completed.

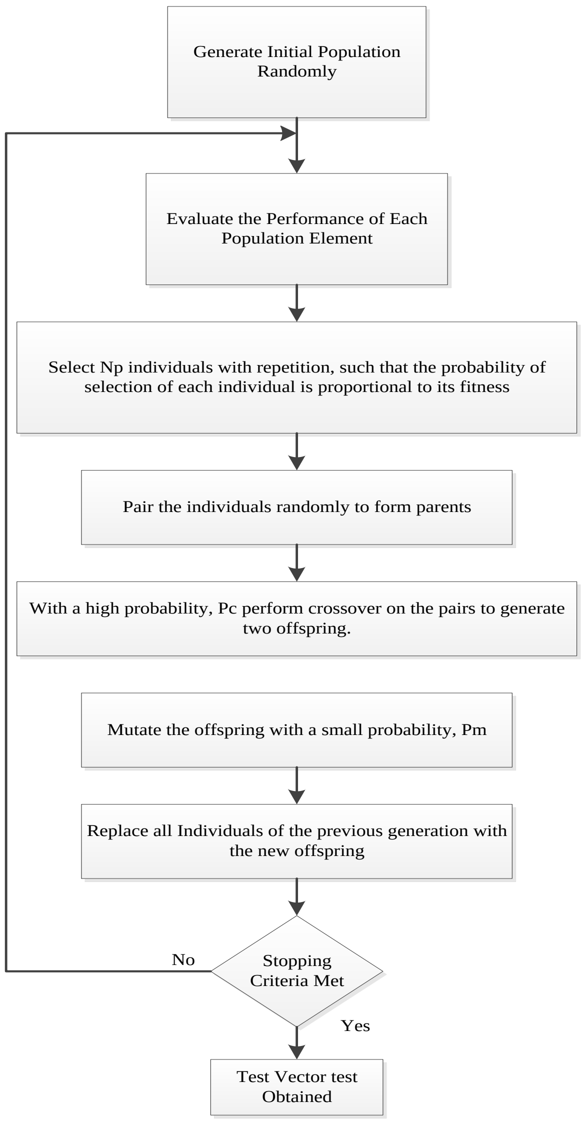
Generate Initial Population Randomly
Evaluate the Performance of Each Population Element
Select Np individuals with repetition, such that the probability of selection of each individual is proportional to its fitness
Pair the individuals randomly to form parents
With a high probability, Pc perform crossover on the pairs to generate two offspring.
Mutate the offspring with a small probability, Pm
Replace all Individuals of the previous generation with the new offspring
No
Stopping Criteria Met
Yes
Test Vector test Obtained

Figure 1. Flow chart showing GA based fault Detection

1.2 Genetic Algorithm in Design Process

Deterministic test generation algorithms for combinational circuits and sequential circuits have been used in the past, but execution times are often long, due to the large number of backtracks that often occur. Simulation-based approaches have also been used and are effective in reducing execution time. However, fault coverage is often lower. The use of GAs for simulation-based test generation has been shown to provide better fault coverage than deterministic algorithms in some cases and lower execution times. A simple GA with non-overlapping populations has been used, due to the irregular search space.

Furthermore, in contrast to other applications, the GA is invoked repeatedly for test generation, and the objective with each invocation is to generate a vector or sequence of vectors to aid in detecting one or more faults. Because of the large number of invocations of the GA and the high cost of accurate fitness evaluation, the GA parameters must be set to reduce the computation time to a reasonable level, at the expense of solution quality. In addition, approximate fitness functions can be used to further reduce the computation time.

GA encoding for the test generation problem has been straightforward. A simple string representation is typically used in which each gene represents the logic value to be applied to a particular input of the circuit at a specific time. The GA contains non-overlapping population of binary coded strings which undergo the evolutionary process of selection, crossover and mutation.

2. PROBLEM FORMULATION

In order to generate a test for a fault condition it is necessary to model the fault condition and simulate the circuit operation with the modeled fault condition present. By using a fault model the physical defect can be represented by altering the characteristics of a circuit to enable it to perform as if a fault condition were present.

Detection of faults in a circuit made of such logic gates can be done by making the individual modules for the gates and then propagating the fault condition to the main output of the circuit. For a given input combination if the true output and the output due to the faulty condition differ then the fault can be easily detected by that input combination.
The procedure involves the application of a sequence of patterns and the states of the Circuit under Test (CUT) advances step by step for time frames according to the applied test vector patterns. Initially the states of the flip-flops of the circuits are at the unknown state “X”. As the patterns are continued to be applied, the states of the flip-flops will gradually be set to the known states ‘1’ or ‘0’ and the faults are detected and dropped by simulated patterns. The elapsed time for first several patterns will be higher than those of the latter due to the fact that many faults are activated and fault effects propagate along the circuits are mainly difficult to detect and their propagation to the primary output is most time consuming. Since for a sequential circuit, there is dependence between test patterns, the state of the circuit depends on previous patterns [9, 10].

3. PROBLEM IMPLEMENTATION

Sequential circuit test generation using deterministic algorithm is a complex process and time consuming. The technique involves the excitation of each target fault and the fault effects need to be propagated to a primary output. Back tracing is a critical step and is used to determine component input values required to obtain a particular output value [11, 12]. A fault simulator is used to select the best vector to apply in each time frame from a set of random vectors. By using GA test vector streams are repeatedly generated in every generation of GA. GA generates each test vector with a random initial population; each bit of an individual string represents the value applied to primary input. The objective function is used to evaluate fitness of each candidate test vector and best vector evolved in each generation is selected to act as an element of final test vector set. The problem specific knowledge has been used in fitness evaluation allowing studying the effectiveness of Gas alone for the test generation application.

When the fault detection process is started during the first few generations we always have the maximum number of faults detected but the number of faults detected drops off with successive vectors. The GA based test generator is shown below as flow chart [13].

The Abstract form of the algorithm is presented below

```
1.  Phase I
    Perform the good circuit simulation i.e. Initialize
    the flipflops with a set of input vectors.
    End;

2.  Phase II
    While (stop ==0 | Remaining faults > 0)
                    Apply the stream of input
    vectors to circuit inputs
        Perform the simulation for the input sequences
        Compare the obtained responses with the true
        responses
    end loop;
    end Phase II;
    end GA;
```

The proposed algorithm consists of two phases. In the first phase, test vectors are generated to initialize the flip-flops. Then test vectors are generated to detect as many faults as possible in phase 2. Hence by first performing the true value simulation for several or group of patterns and then followed by performing the fault simulation for the latter patterns, the time consuming faults can be dropped in the initial runs and thereby saving the total simulation time [14].

The main purpose of doing phase I simulation is to initialize the flip-flops so the fitness of test vector can be found by considering the number of flip-flops set to a known state. So the vector, which causes an increase in fraction of flip-flops set or reset, is awarded higher fitness value. It can also be called as a good circuit simulation. In phase II, the objective is to detect the maximum number of faults as possible so the fitness of the test vector in this phase is a measure of number of previously undetected faults in previous generations. In order to have fault detection at the output the propagation of fault effects is to be accomplished [15, 16].

The overall computation effort is depending heavily on fitness evaluation function. The GA operations (selection, crossover and mutation) are not very time consuming. The computation time can be improved by using a less accurate fitness function which is easy to evaluate but this technique has an adverse effect on the quality of the simulation i.e. it may result in more test vectors and lower fault coverage.
A flow chart showing the steps involved in the fault simulation process is shown below

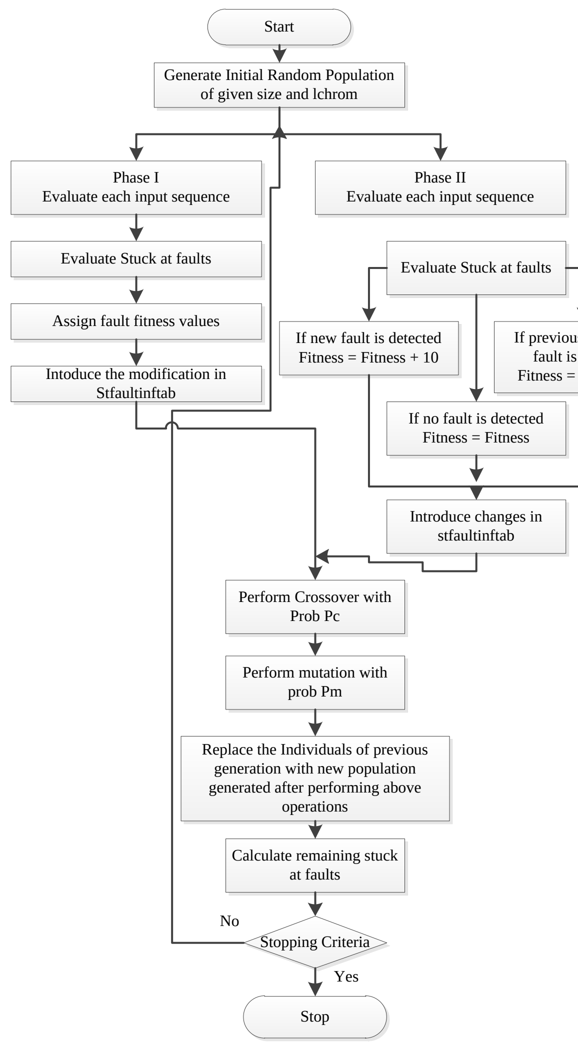
Start
Generate Initial Random Population of given size and lchrom
Phase I Evaluate each input sequence
Phase II Evaluate each input sequence
Evaluate Stuck at faults
Assign fault fitness values
Intoduce the modification in Stfaultinftab
Evaluate Stuck at faults
If new fault is detected Fitness = Fitness + 10
If previously detected fault is detected Fitness = Fitness + 1
If no fault is detected Fitness = Fitness
Introduce changes in stfaultinftab
Perform Crossover with Prob Pc
Perform mutation with prob Pm
Replace the Individuals of previous generation with new population generated after performing above operations
Calculate remaining stuck at faults
No
Stopping Criteria
Yes
Stop

Figure 2. Flow Chart Showing the Detection Procedure

The two phase fault simulation technique has been successfully applied over two sequential circuits namely 4 bit parallel load Shift Register and Vending Machine Controller.

Test Circuit First (TCI): 4 Bit Parallel Load Shift Register

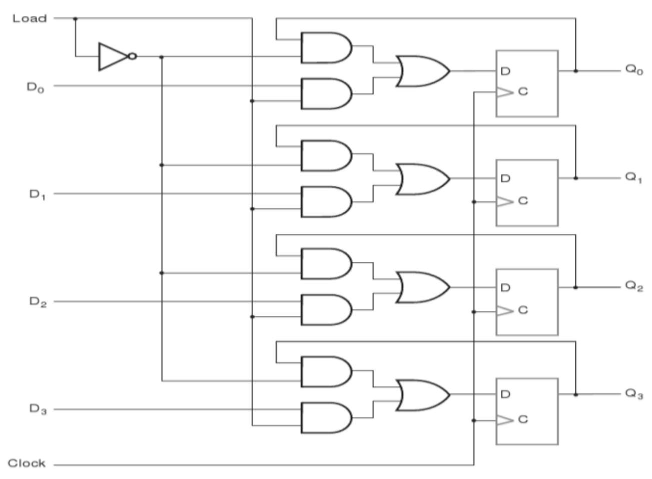

Figure 3. 4 bit Parallel load Shift Register

Statistics: 4 inputs; 4 output; 13 gates excluding flip-flop gates.

Function: It will allow having a parallel load of inputs to be transferred at the output.

The circuit has been fully evaluated for the possible stuck at faults instances and the proposed simulator produces complete fault coverage.

Test Circuit Second (TCII): Vending Machine Controller

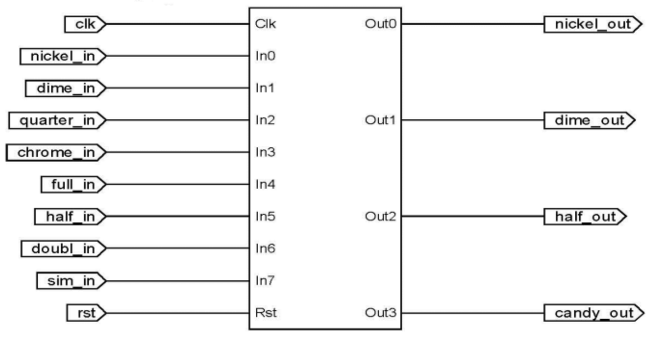

Figure 4. Vending Machine Controller: Top view

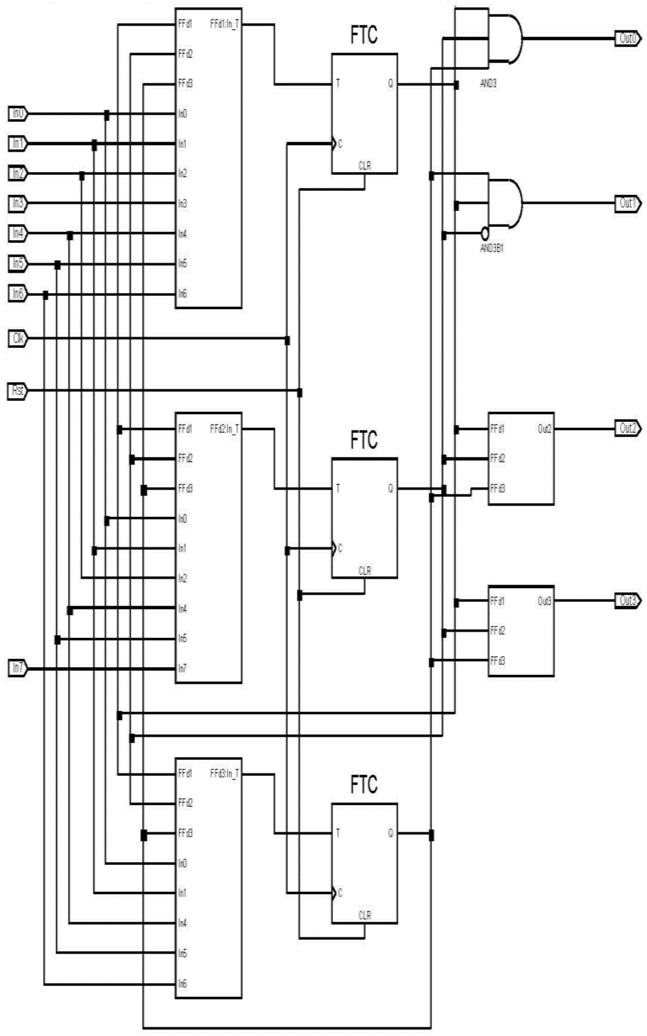

Figure 5. Vending Machine Controller Block View

Statistics: 9 inputs; 4 outputs; 51 logic gates excluding gates in flip-flops

Function: It is a state machine and allows having different output states for different input combinations.

The circuit has been fully evaluated for the possible stuck at faults instances and the proposed simulator produces complete fault coverage.

4. EXPERIMENTAL RESULTS AND DISCUSSION

The technique has been successfully applied to detect faults in two test circuits. It has been shown with the help of figures below that complete fault coverage is achieved for test circuits considered. Apart from this some experiments are also performed to find out the effect of varying various GA parameters given below on the detection rate. The Parameters used for analysis are: Population size: Pop size, Probability of Crossover: Pc, Probability of Mutation: Pm, Length of the chromosome: Lchrom, Maximum number of generations: Maxg, Fitness Award Value: AN and Type of Crossover (CT): 1point (1pt), 2point (2pt), Uniform crossover (uc)

4.1 Plots for 4 bit Parallel Register(Test Circuit I)

(a) Effect of Population Size:

Specifications: Pop size: 3,6,10,12,15, Pc: 0.6, Pm: 0.25, lchrom: 5,Maxg: 30,AN: 1:10,CT: 2pt and Stuck fault remaining: 6,3,0,0

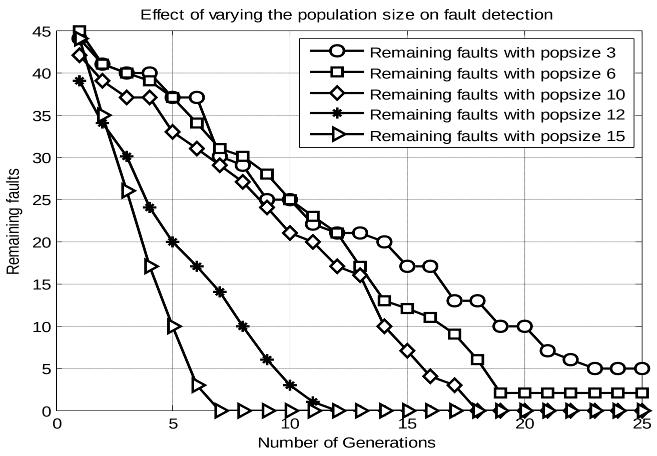

Figure 6. Effect of changing population size on fault detection

The figure 6 shows that if we vary the population size parameter of GA it will have a direct effect on the number of iterations used to have a full fault coverage. In order to reduce the number of Generations we have to increas the population size. The increased population size will basically gives a probability of starting with some good some solutions without having a need to generate and to converge to them in future iterations and it has a direct impact on the number of iterations required to have a complete fault detections .

(b) Effect of Maximum Number of Generations:
Specifications: Pop size: 4, Pc: 0.6, Pm: 0.25, lchrom:5,Maxg: 5,10,15,20,AN:1:10,CT:2pt and Stuck fault remaining: 19, 10, 3, 0

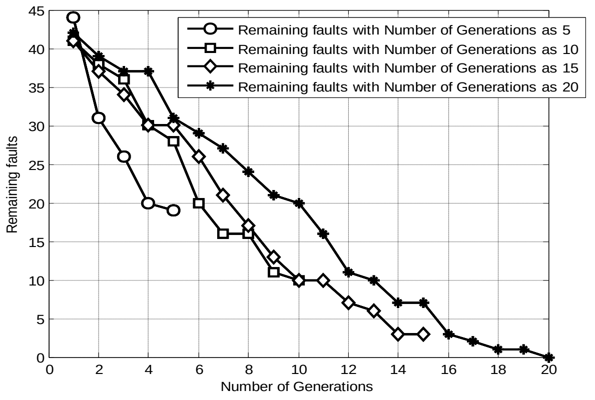

Figure 7. Effect of changing the number of generations on fault detection.

The figure 7 shows that if we don't change the GA parameters like Pc, Pm and Popsize. In order to have a full fault coverage the algorithm have to be run for extra runs to explore the search space for the undetected faults. It has been found that without having a chance of optimizing the GA parameters during course of fault detection the GA has to be re run for greater number of loops to find a better solution. However , with the help of later graphs it has been prooved that with an intelligent optimization of some GA parameters the algorithm can be converged to a good solutions with less number of iterations required.

(c) Effect of Probability of Crossover:
Specifications: Popsize: 7, Pc: 0.4,0.5,0.55,0.65,0.8, Pm: 0.1, lchrom: 5,Maxg: 25,AN: 1:10,CT: 2pt and Stuck fault remaining: 6, 4,2,0,0

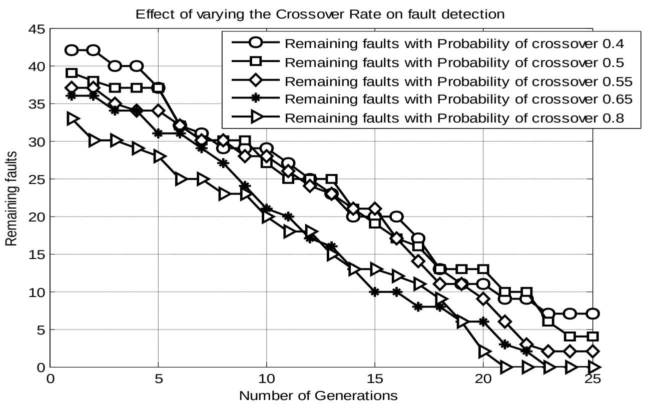

Figure 8. Effect of varying the crossover rate

The figure 8 shows that the probability of crossover will need to be higher if we want to have a faster convergence to the solution. The probability of crossover has a direct impact on the quality of future generations, if we start with a higher crossover probability or if we modify during the course of run then there are better chances that good qualities of some better parents will be transferred to future mating pools and it will help to achieve good detection rates.

(d) Effect of Probability of Mutation:
Specifications: Popsize: 7, Pc: 0.6, Pm: 0.01,0.1,0.3, lchrom: 5,Maxg: 25,AN: 1:10,CT: 2pt and Stuck fault remaining: 8, 3,0

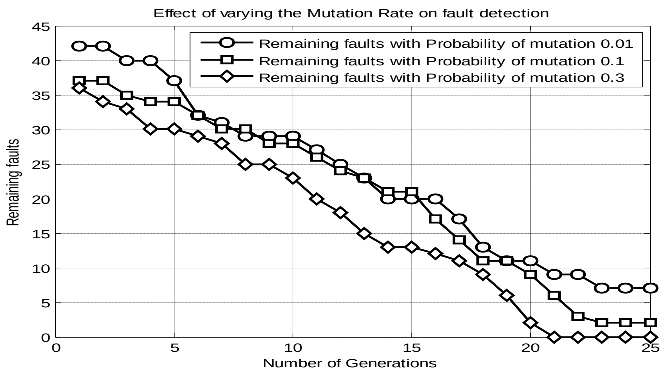

Figure 9. Effect of varing mutation rate on fault detection

The figure 9 clearly demonstrate that by merely a reversal of a bit in a vector stream may lead to generationto new test sequences to have better fault coverage in lesser number of generations. Mutation basically helps to have some alteration in the basic character of a test vector which otherwise would be difficult to achieve by crossover operator. It has been tried that by selectively manipulating the bit patterns the fitness value of a test vector can be enhanced.

(e) Effect of Fitness Award Value:
Specifications: Pop size: 7, Pc: 0.6, Pm: 0.1, lchrom: 5,Maxg: 15,AN: 1:2,1:4,1:8,1:10,CT: 2pt and Stuck fault remaining: 8,4,2,0

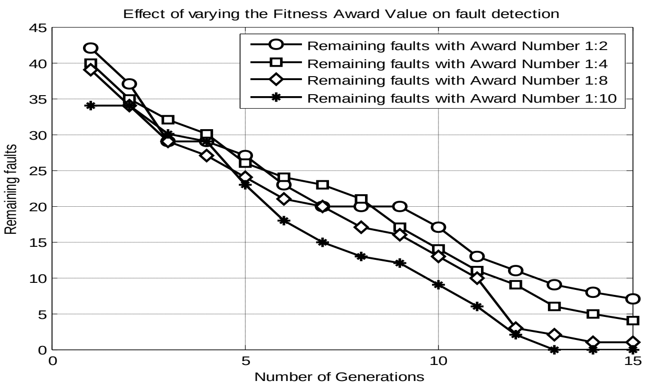

Figure 10. Effect of varying fault detection value on number of generations.

The figure 10 have been plotted to illustrate the effect of fitness award value on the number of iterations required. It has been clearly shown that as the algorithm progresses if we award a higher fitness value to those test vectors which have detected newer faults as compared to the already detected faults then we may have a better solution is fewer generations.

(f) Effect of Adaptive Crossover and Mutation Rate:
Specifications: Pop size: 7, Pc: 0.4 -0.8, Pm: 0.01-0.3, lchrom:5, Maxg: 25,AN:1:2,CT:2pt and Stuck fault remaining: 0

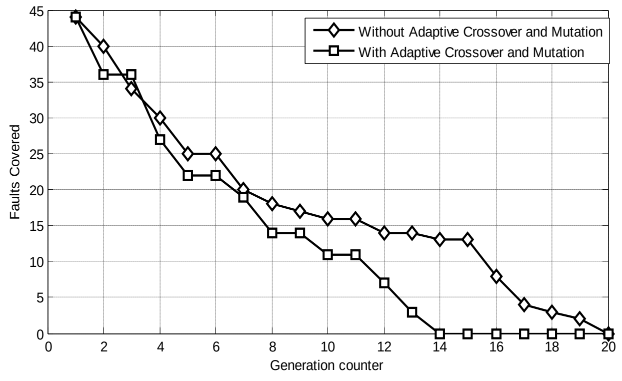

Figure 11. Effect of varying crossover and mutation rate on number of generations.

By using Adaptive Crossove and Mutation rate we can actually allow the GA to explore the newer regions of the search space and hence will prevent it to be caught in local minima. In the algorithmic implementation by varying these rates during the course of the run newer individuals will have a better chance of generation.

(g) Effect of Different Crossover Techniques
Specifications: Pop size: 7, Pc: 0.5 , Pm: 0.1, lchrom:5,Maxg: 25,AN:1:2,CT:1pt,2pt,uc and Stuck fault remaining: 0

The figure has been plotted to demonstrate that it is possible to change the number of generations required for full fault detection by merely changing the crossover style.

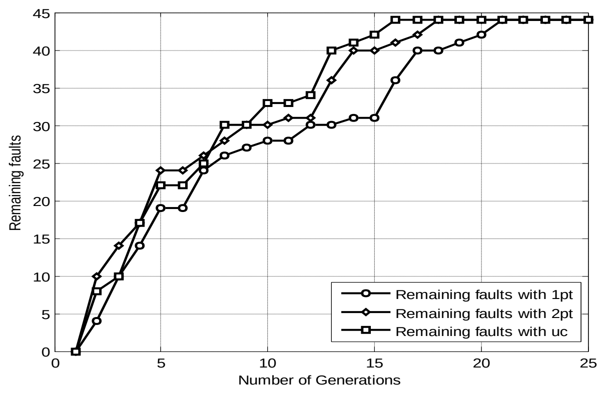

Figure 12. Different crossover rates used and their effect on rate of fault detection.

Figure 12 clearly shows that different crossover techniques require different number of iterations to have complete fault coverage. It has been found that instead of following a fixed approach to have a crossover between two parents more realistic results will be obtained if some rule based technique is adopted to have a offspring generation just like what has been shown by the uniform crossover plot.

4.2 Plots Vending Machine Controller (Test Circuit II)

(a) Effect of Population Size:
Specifications: Pop size: 10,20,30,50, Pc: 0.6, Pm: 0.25, lchrom: 8, Maxg: 30, AN: 1:10, CT: 2pt and Stuck fault remaining: 19,9,0,0

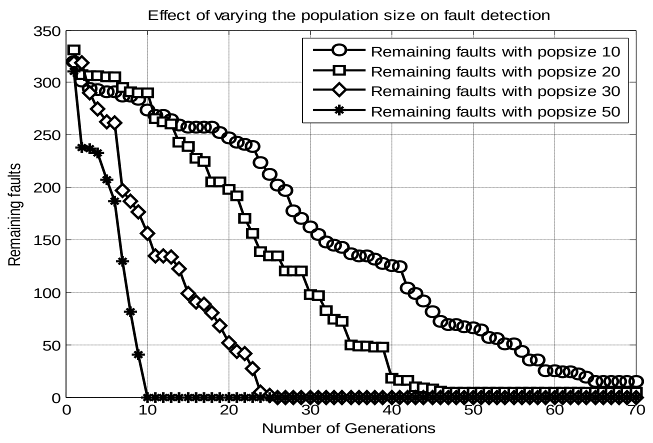

Figure 13. Effect of varying population size on fault detection rate.

The effect of varying the population size on the number of generations required is clearly demonstrated by the above figures. By increasing the population size the GA will find more number of test vectors to operate with and stands a good chance to quickly converging to good solution in lesser time.

(b) Effect of Maximum Number of Generations:
Specifications: Popsize: 8, Pc: 0.5, Pm: 0.15, lchrom: 8,Maxg: 19,45,80,AN: 1:10,CT: 2pt and Stuck fault remaining: 40,19,0

The effect of varying the number of generations on the totals faults detected in that time frame is shown in the figure 14.

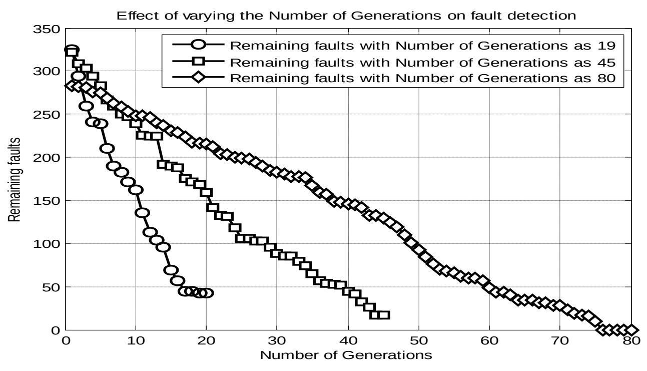

Figure 14. Effect of varying the number of generations on fault detection

It has been depicted that if the GA is allowed to run for greater number of generations then without even changing the GA parameters it will converge to a good solution.

(c) Effect of Probability of Crossover:

Specifications: Popsize: 15, Pc: 0.4,0.6,0.8, Pm: 0.1, lchrom: 8,Maxg: 30,AN: 1:10,CT: 2pt and Stuck fault remaining: 18, 7, 0

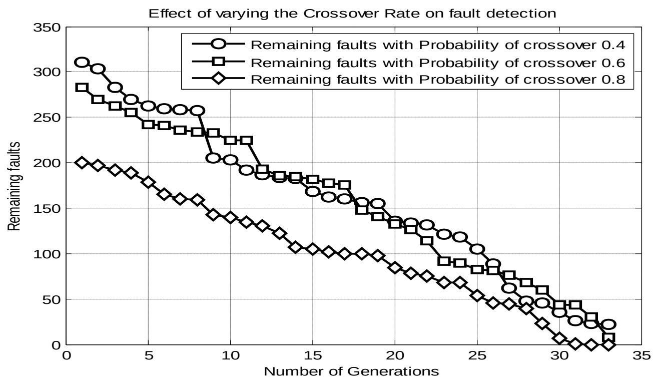

Figure 15. Effect of varying probability of crossover on fault detection rate.

The figure 15 have been plotted to show that by varying the crossover probability there is a great effect on number of generations required. Depending on the number of crossover performed in a GA, it has been found that parents with good fitness values stand a good chance of transferring their properties to greater number of offspring's and lesser are the chances of losing their goodness in future generations. But care needs to be taken that the probability of crossover cannot be increased to maximum as it may lead GA to a temporary bad solution.

(d) Effect of Probability of Mutation:

Specifications: Popsize: 14, Pc: 0.6, Pm: 0.01,0.1,0.3, lchrom: 8,Maxg: 35,AN: 1:10,CT: 2pt and Stuck fault remaining: 25, 9, 0

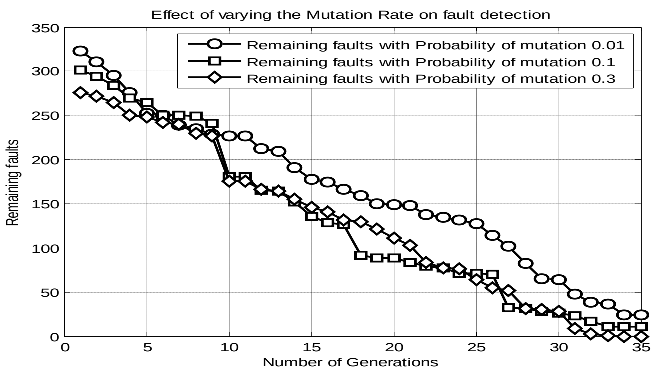

Figure 16. Effect of varying the mutation rate on fault detection.

The effect of varying mutation rate on the fault detection is clearly shown in the figure 16. Probability of mutation will try to introduce some new qualities in the nature of test vectors and sometimes it may lead to the generation of test vectors which are difficult to be generated by only adopting other GA operators, but care must be taken that it should not be used with greater values as it may lead to loss of overall quality of solution.

(e) Effect of Fitness Award Value:

Specifications: Popsize: 20, Pc: 0.6, Pm: 0.1, lchrom:8,Maxg: 30,AN:1:2,1:10,1:20,CT:2pt and Stuck fault remaining: 12,0,0

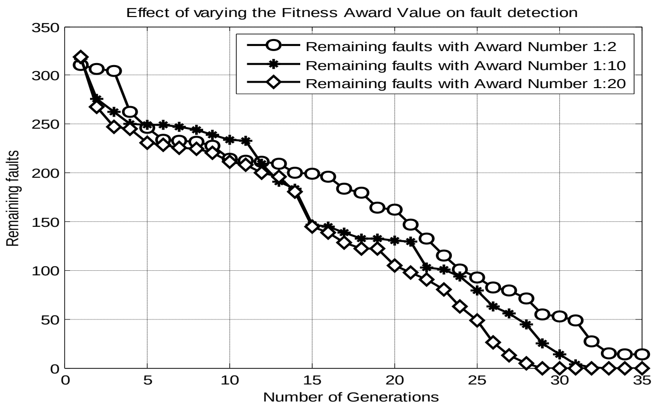

Figure 17. Effect of varying the fitness award value on fault detection rate.

The figure 17 shows that by changing the fitness award value to test vectors we can change the total number of iterations required for fault detection. The fitness award value helps in discrimination those test vectors which actually detect new undetected faults, so by assigning them greater values with respect to the total number of faults detected it has been tried to give them a better chance of survival to coming generations to be tested for fault detection.

(f) Effect of Adaptive Crossover and Mutation Rate:

Specifications: Popsize: 15, Pc: 0.4 -0.8, Pm: 0.01-0.3, lchrom: 8, Maxg: 35, AN: 1:2, CT: 2pt and Stuck fault remaining: 0

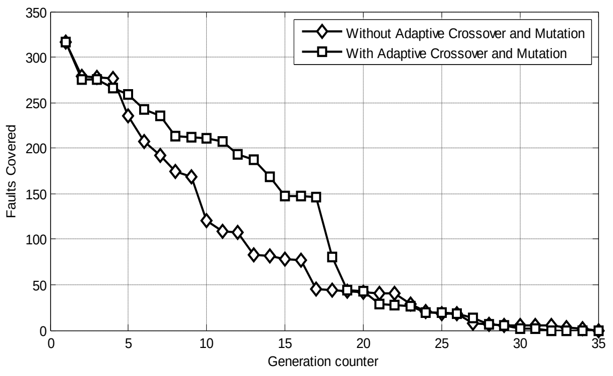

Figure 18 . Effect of varying crossover and mutation rate during fault detection process.

It has been shown in the figure 18 that the adaptive crossover and mutation rates can actually change the total fault detection rate. It has been found that something the GA may not be able to increase the number of faults detected in some iterations at all so by rapidly monitoring the fault detection rate during the course of run we modify some GA parameters to help GA to explore some new regions of solution space so as to detect those faults which have not been detected yet.

(g) Effect of Different Crossover Techniques

Specifications: Popsize: 20, Pc: 0.5 , Pm: 0.1, lchrom:8,Maxg: 40,AN:1:2,CT:1pt,2pt,uc and Stuck fault remaining: 0

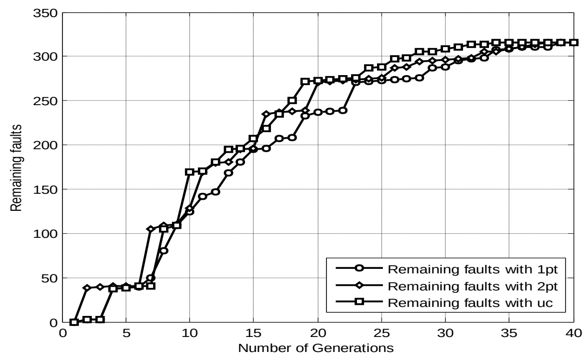

Figure 19. Different crossover techniques and its effect on fault detection process.

If we adopt different crossover strategies then it may affect the total fault detection rate as it may lead to new individual generation with better fault detection possibilities. It has been found that the crossover should be performed by using some heuristics so as to have some new generations possible which may lead to good fault detections as has been shown with the help of uniform crossover plot.

5. CONCLUSION

In this paper a 2-phase fault simulator has been proposed with this technique a true simulation followed by fault simulation is performed to identify the possible stuck at fault instances. Simulation based techniques can be effectively used for sequential circuits. Genetic algorithm provides a probabilistic based selection and random search of points in a large solution space. The technique adopted has been successfully applied to the two sequential circuits and all faults have been detected. It has been found that instead of concentrating on fixed crossover and mutation rates if variable assignment is used then the detection rate can be considerably increased. For both of our test circuits the GA produced effective test sets with high fault coverage. The GA based test generator is not limited to single stuck at fault model and other fault models can be accommodated with appropriate fitness functions.

6. REFERENCES

[1] Michael S. Hsiao, Elizabeth M. Rudnick, Janak H. Patel, “ Sequential circuit test generation using dynamic state traversal”, Proc of European Design and Test Conference, ED&TC 97, pp 22 – 28, 1997

[2] Pomeranz, I. Reddy, S.M., “On maximizing the fault coverage for a given test length limit in a synchronous sequential circuit”, Proc of VLSI Test Symposium, pp 173 – 178, 2003

[3] Pomeranz, I. Reddy, S.M., “TOV: Sequential Test Generation by Ordering of Test Vectors”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp 454 – 465, 2010

[4] Wen Ching Wu, Chung Len Lee, "A Two-Phase Fault Simulation Scheme for Sequential Circuits," pp.41, 10th Anniversary Compendium of Papers from Asian Test Symposium 1992-2001 (ATScomp'01), 2001

[5] Rudnick, E.M.; Patel, J.H.; Greenstein, G.S.; Niermann, T.M., “A genetic algorithm framework for test generation”, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, pp 1034 – 1044, 1997

[6] Pomeranz, I. Reddy, S.M., “Vector replacement to improve static-test compaction for synchronous sequential circuits”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp 336 – 342, 2001

[7] Elizabeth M. Rudnick,Janak H. Patel,Gary S. Greenstein,Thomas M. Niermann, “Sequential circuit test generation in a genetic algorithm framework”,Proceedings of the 31st annual Design Automation Conference, pp 698-704, 1994

[8] E. M. Rudnick, J. G. Holm, D. G. Saab, and J. H. Patel, "Application of simple genetic algorithms to sequential circuit test generation," Proc. European Design and Test Conf., 1994, pp. 40-45.

[9] M. Srinivas and L. M. Patnaik, "A simulation-based test generation scheme using genetic algorithms," Proc. Int. Conf. VLSI Design, pp. 132-135, 1993

[10] Pomeranz, I. Reddy, S.M., “ A built-in self-test method for diagnosis of synchronous sequential circuits”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, pp 290 – 296, 2001

[11] Shuo Sheng Hsiao, M.S., “Efficient sequential test generation based on logic simulation”, IEEE Transactions On Design & Test of Computers, pp 56 – 64, 2002

[12] Xiaoming Yuy, Jue Wuz, Elizabeth M. Rudnicky, “Diagnostic Test Generation for Sequential Circuits”, Proc of International Test Conference 2000 (ITC'00), pp.225-234, 2000

[13] V.Rajesh, Ajai Jain, “ Automatic Test Pattern Generation for Sequential Circuits using Genetic Algorithm”, Proc of Eleventh International Conference on VLSI Design: VLSI for Signal Processing, pp 270-273, 1998

[14] Xiaoming Yu Abramovici, M., “Sequential circuit ATPG using combinational algorithms”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp 1294 – 1310, 2005

[15] Michael S. Hsiao, Elizabeth M.Rudnick and Janak H. Patel, “Fast static compaction algorithms for sequential circuit test vectors, IEEE Transactions on Computers, pp 311 – 322, 1999

[16] Ruifeng Guo,Sudhakar M. Reddy ,Irith Pomeranz, “ On Improving a Fault Simulation Based Test Generator for Synchronous Sequential Circuits”, Proceedings of the 10th Asian Test Symposium table of contents, pp 82-87, 2001.