International Journal of Electronics Signals and Systems

Volume 1 | Issue 2 Article 13

September 2011

Generating Test Patterns for Multiple Fault Detection in VLSI Circuits using Genetic Algorithm

R, Balasubramanian
Department of ECE, Amrita School of Engineering, Ettimadai, Coimbatore., rbala1990@gmail.com

Available online at www.interscience.in

Generating Test Patterns for Multiple Fault Detection in VLSI Circuits using Genetic Algorithm

Balasubramanian R, Goutham K, Jenitha S, Rahul R M, Thirupathie Raja B, Vijaya Kumar S

Department of ECE, Amrita School of Engineering, Ettimadai, Coimbatore.
Email: rbala1990@gmail.com

Abstract **-** In this paper we propose a method for the automatic test pattern generation for detecting multiple stuck-at-faults in combinational VLSI circuits using genetic algorithm (GA). Derivation of minimal test sets helps to reduce the post-production cost of testing combinational circuits. The GA proves to be an effective algorithm in finding optimum number of test patterns from the highly complex problem space. The paper describes the GA and results obtained for the ISCAS 1989 benchmark circuits.

Keywords— *genetic algorithm, multiple stuck-at-faults, test pattern generation*

I. INTRODUCTION

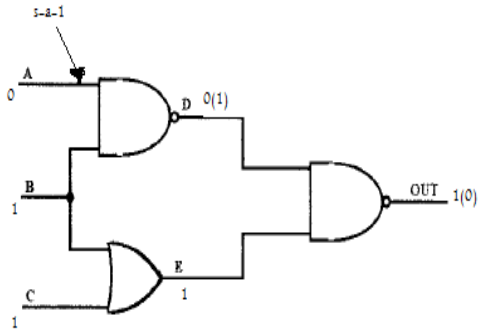
In recent years, Integrated circuits (ICs) produced are extremely intricate and are ever increasing in complexity, as they are required for use in various applications [1]. Reliable and correct operations of ICs are of very important concern as they may be used in medical, military or flight control systems. To ensure the reliability of such ICs, it is important to test their performance to detect any defects prior to using them in a fully operational environment. Thus testing of VLSI circuits is an important concern for the chip manufacturers. Testing of VLSI circuits is achieved by generating test-patterns with high fault coverage, which on the other hand should be a minimal test set in order to reduce the total product cost by saving time.

Genetic Algorithms (GAs) are a stochastic global search method that mimics the metaphor of natural biological evolution. Based loosely upon the Darwinian ideas of evolution and natural selection [2], they have proven themselves to be practical, robust optimisation tools, well suited to the problem of deriving optimal test sets. They have been applied successfully in a variety of VLSI design context [3-4]. In [5] the author has generated test patterns for VLSI circuits using Genetic Algorithm for both delay fault and single stuck-at-fault. Here in this paper we have implemented GA using C language for generating minimal test set, used for detection of multiple stuck-at-faults in VLSI combinational circuits.

2. FAULT MODEL

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 modelled fault condition present. With the help of a fault model the physical defect can be represented by changing the characteristics of a circuit to enable it to perform as if a fault condition were present. The most common model used for logical faults is the single stuck-at-fault model [6]. It assumes that a fault in a logic gate results in one of its inputs or the output being fixed to either a logic 0 (stuck-at-0) or a logic 1 (stuck-at-1). Stuck-at-0 and stuck-at-1 are often abbreviated to s-a-0 and s-a-1 respectively. Let us consider the test circuit in figure 1 with input A s-a-1. The NAND gate perceives the input A as logic 1 irrespective of the logic value placed on the input. The output of the circuit in the figure 1 is logic 1 for the input pattern shown, when the s-a-1 fault is present. The fault-free output of the circuit is logic 0. Therefore, the pattern shown in the figure 1 can be used as a test for the A input s-a-1, since there is a difference between the output of the fault-free and the faulty gate.

Fig. 1.Single Stuck-at-fault



The stuck-at-fault model is also used to represent multiple faults in circuits. In a multiple stuck-at-fault model it is assumed that more than one signal line in the

circuit are stuck-at logic 1 or 0; in other words a group of stuck-at faults exist in the circuit at the same time. A variation of the multiple faults is the unidirectional fault. A multiple fault is unidirectional if all of its constituent faults are either s-a-0 or s-a-1 but not both simultaneously. The stuck-at model has gained wide acceptance in the past mainly because of its relative success with small scale integration.

In a given circuit,

Number of possible single stuck-at-fault = 2*n

Number of possible multiple stuck-at-fault = 3^n -1

Where n is the number of nodes/signal lines in the circuit under test[7-8]. Another important parameter is the fault coverage[8], it gives the information about how many faults can be detected using the test patterns generated. It is given by the relation

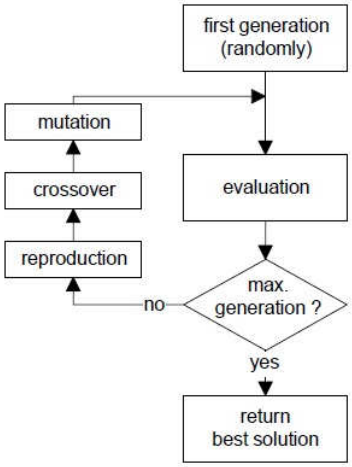
Fault coverage = 100 * (number of faults detected by the test pattern / total faults in circuit under test)

3. GENETIC ALGORITHM

Genetic Algorithms (GAs) draw inspiration from the biological and evolutionary processes of selection, crossover and mutation. GAs use probabilistic search techniques to find globally optimal solutions in large, complex search spaces. The process of GAs is presented in Figure 2.

GAs contains populations of N candidate solutions called chromosomes. Each bit in a chromosome is referred to as a gene. Associated with each chromosome is a fitness value depending on the application, which rates its competence as a solution. Starting the process with an initial population called parent chromosomes which is randomly generated, the offspring chromosomes evolve by crossover and mutation operations over parent chromosomes in the subsequent generations. The fitter offspring chromosomes replace the less fit parent chromosome and they themselves become the parent chromosomes for the next generation. This process of replacing chromosomes is called reproduction which increases the average fitness of the population. This entire process is iterated till the terminating condition is reached. The terminating condition may be maximum number of iterations or maximum fitness value is attained.

Fig 2. GA flow chart



4. IMPLEMENTATION

An initial population of 10 chromosomes for smaller circuits and 25 chromosomes for larger circuits are generated randomly. Now we give the initial test set as input to the circuit under test. Initially we take a fault free response of the circuit. Then we simulate stuck at faults and see whether the obtained output differs from the fault free response. If there is any change in the response then we can say that the particular test pattern can detect that fault. Hence we increase the fitness of the chromosome. By repeatedly doing this process for all possible fault combinations in a given circuit under test, we could get the fitness for each of the chromosome that was generated in a random fashion. Then we get the best fit from the initial population.

The fitness function is given by the formula,

Fitness = $\sum^{\text{Combination of faults}}$ Number of faults simulated*[Detected? (0 or 1)]

After selecting the best fit from the initial population we do genetic operations such as crossover and mutation for generating a new set of patterns. We find the fitness of the new off-springs that were formed from genetic operations. The off-spring chromosome replaces the parent chromosome if its fitness value is greater than that of parent chromosome. Thus each chromosome has a fitness value and genetic algorithm finds the chromosome with maximum fitness value.

The crossover used is single point crossover where two random chromosomes are taken, a random crossover point is fixed, and the properties of chromosomes are interchanged. Mutation is done by selecting a random gene from a random chromosome and the value is

inverted. The probability of crossover is set as 0.9 and probability of mutation is set as 0.1.

The pseudo-code for the model is given as follows:

Read the circuit file

Initialize population randomly for Genetic Algorithm

{

{

{

Simulate the circuit to find fault free and faulty response of the circuit for the chromosome

} Loop till all combination of faults are simulated

Find the fitness of the chromosome

} Loop to find fitness of all chromosomes

Reproduce using fitness values calculated

Check for stopping condition

Crossover with a probability of 0.9

Mutate with a probability of 0.1

} Loop till max. iteration is reached or max. fitness is reached

5. RESULTS

The results are tabulated for different sets of faults and different ISCAS 1989 benchmark circuits.

TABLE 1

Benchmark Circuit	Number of faults injected (max. 10)	Fault coverage (in %)	Number of iterations taken to converge (max. 100)
C17	3	100	2
C432	6	100	18
C880	9	100	27
C1355	10	90	100

C17 Benchmark circuit

INPUT(G1)

INPUT(G2)

INPUT(G3)

INPUT(G4)

INPUT(G5)

OUTPUT(G16)

OUTPUT(G17)

G8 = NAND(G1, G3)

G9 = NAND(G3, G4)

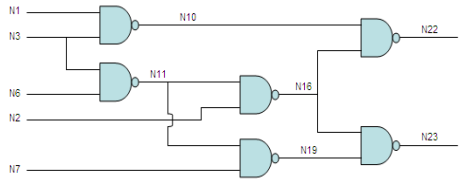
G12 = NAND(G2, G9)

G15 = NAND(G9, G5)

G16 = NAND(G8, G12)

G17 = NAND(G12, G15)

Fig. 3 C17 Benchmark Circuit



For example for the given c17 circuit 3 faults were simulated namely nodeG1 s-a-1 , nodeG8 s-a-0, nodeG16 s-a-1

The following fittest chromosome was obtained 01110 which would detect all the above mentioned 3 faults in all combinations. The chromosome 0-1-1-1-0 are the inputs to the nodes G1, G2, G3, G4 and G5 respectively.

We can see that as the size of the circuit increases the genetic algorithm takes more time to find the optimal test pattern. Also in C1355 circuit the genetic algorithm cannot find a test pattern that can detect all the faults simultaneously in 100 iterations.

6. CONCLUSION

In this paper, we find that genetic algorithm helps us in finding out quickly an optimal set of test patterns that can detect the given set of faults for a circuit. This algorithm can be used in automatic test pattern generation tools to generate test patterns to detect the given multiple stuck at faults. Further improvement can be done by including delay faults and implementing it for sequential circuits.

REFERENCES

[1] A. Breuer and A. D. Friedman, “Diagnosis and reliable design of digital systems”, Science Press Inc.1977.

[2] Holland J.H, “Adaption in Natural and Artificial systems”, The MIT Press,Cambridge,USA,1996.

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

[4] Srinivas M., Patnaik, L.M., "A simulation based test generation scheme using genetic algorithms", 6th International Conference on VLSI design, 1993, pp132-135.

[5] O'Dare M.J., Arslan T., "Generating test patterns for VLSI circuits using a genetic algorithm", Electronic Letters, May 1994, Vol.30,No.10,pp 778-779

[6] S. Park and M.R. Mercer, "An Efficient Test Generation System for Combinational Logic Circuits", IEEE Trans. On Computer-Aided Design. Vol11, No7, July 1992,pp 926-940.

[7] M.J. O'Dare and T. Arslan, "Generating test patterns for VLSl circuits using a genetic algorithm", ELECTRONICS LETTERS, Vol. 30, No. 10, 12th May 1994.

[8] Enrico Macii , Tara Wolf, "Multiple Fault Diagnosis in Combinational Networks", 0-7803-2428-5195, 1995 IEEE.

[9] Arslan T., O'Dare M.J., "A Genetic Algorithm for multiple fault model test generation for combinational VLSI circuits", GALESIA 97, Conf.Publ.No.446, sep 1997, pp462-466.

❑❑❑