

Implementation of Compaction and Genetic Algorithm for ATPG Generated Partially Specified Test Data

¹N.Vani; ²Dr. P. Venkateswarlu & ³Md. Afzal

¹M.Tech (CSE), Department of Computer Science & Engineering Nagole Institute of Technology & Science, Kuntloor (V), Hayathnagar (M), RR District, Hyderabad, India.

E-mail id: vaniyamini@gmail.com

²Professor & HOD, Department of Computer Science & Engineering.

E-mail id: venkat123.pedakolmi@gmail.com

³Assistant Professor, Department of Computer Science & Engineering.

E-mail id: afzal.rubeena@gmail.com

Abstract:

In this paper the ATPG is implemented using C++. This ATPG is based on fault equivalence concept in which the number of faults gets reduced before compaction method and also used genetic algorithm for random test pattern generation. This ATPG uses the line justification and error propagation to find the test vectors for reduced fault set with the aid of controllability and observability and genetic algorithm solves many search and optimization problem effectively. Single stuck at fault model is considered. The programs are developed for fault equivalence method, controllability Observability, automatic test pattern generation and test data compaction using object oriented language C++. . Experiment results showed that the genetic algorithm improved the ability of global search and increases the fault coverage.

Keywords:

Automatic Test Pattern Generation (ATPG); Genetic Algorithm (GA); Design Validation; Evolutionary Algorithm; Test vector; compaction; ISCAS; ATPG

1 INTRODUCTION:

In present scenario, digital systems are extremely intricate and increasing in complexity, which are required for use in widening range of domestic and industrial application. So to ensure reliability of these digital circuits, it is necessary

to test their performance to identify any defects prior to using them in a fully operational environment. These circuits are tested by test vectors. The test vectors are generated by efficient automatic test pattern generator (ATPG).The generation of test pattern with high fault coverage rate is a very expensive process for large circuits. An efficient ATPG tool reduces the test pattern generation time and cost, beside the high fault coverage rate. There are many approaches for ATPG, like deterministic approach, simulators etc.The aim of this technique should be both to reduce execution time and to improve fault coverage. Genetic algorithm described by Goldberg[13] is specially suited to solve large scale combination optimization problem.GA have been successfully applied in different areas of VLSI design ,especially in test branches such as test pattern generation [5]. Test vector size is the big issue in the today's technology. As size of circuit increases the size of test vector also increases so that the memory. This paper includes such ATPG which itself compacts test patterns before further compression of the test patterns. ATPG (acronym for both Automatic Test Pattern Generation and Automatic Test Pattern Generator) is an electronic design automation method/technology used to find an input (or test) sequence that, when applied to a digital circuit, enables testers to distinguish

between the correct circuit behavior and the faulty circuit behavior caused by defects[1]. The generated patterns are used to test semiconductor devices after manufacture, and in some cases to assist with determining the cause of failure (failure analysis) the effectiveness of ATPG is measured by the amount of modeled defects, or fault models, that are detected and the number of generated patterns. These metrics generally indicate test quality (higher with more fault detections) and test application time (higher with more patterns). ATPG efficiency is another important consideration. It is influenced by the fault model under consideration, the type of circuit under test (full scan, synchronous sequential, or asynchronous sequential), the level of abstraction used to represent the circuit under test (gate, register-transistor, switch), and the required test quality [2]. The single stuck-at-fault model has been widely accepted as a standard target model to generate a set of test patterns to detect all the stuck faults in the circuit. A single stuck-at fault represents a line in the circuit that is fixed to logic value 0 or 1. The single-stuck fault model is also referred to as the classical or standard fault model because it has been the first and the most widely studied and used. Although its validity is not universal, its usefulness results from the following attributes: The single stuck at fault can be used to represent short or open, caused due to short between ground or power line, causing a signal line remain at a fixed voltage level. If we consider single stuck at fault then the number of faults is $2n$, where n is number of net .In this case we have to find $2n$ test vectors, for each fault (stuck at 0, stuck at 1) on each net. Size of test vector becomes large for large combinational circuits. ATE (Automatic Test Equipment) bandwidth problem cause to handle these test vectors and testing time may be more in this case. No doubt test vector compression methods are available, prior to that, before test generation the number of faults can be reduced sothat the test vectors. The number of faults can be reduced using fault equivalence method and

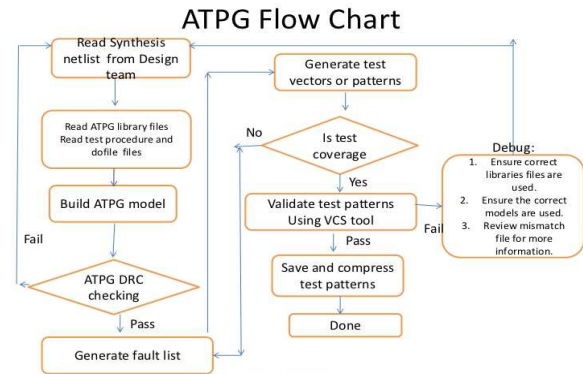
fault dominance method. In this paper an attempt is made to reduce the fault set using fault equivalence method and developed in C++. The logic and flow chart of the program are given in this paper. The results for ISCAS C17 benchmark circuit were analyzed. The generated test pattern for ISCAS C17 circuit is discussed in result section .The logic and flow chart is discussed in this paper. The test pattern compaction algorithm is discussed along with the results and comparisons.

2. ATPG

A defect is an error introduced into a device during the manufacturing process. A fault model is a mathematical description of how a defect alters design behavior. A fault is said to be detected by a test pattern if, when applying the pattern to the design, any logic value observed at one or more of the circuit's primary outputs differs between the original design and the design with the fault. The ATPG process for a targeted fault consists of two phases: fault activation and fault propagation. Fault activation establishes a signal value at the fault model site that is opposite of the value produced by the fault model. Fault propagation moves the resulting signal value, or fault effect, forward by sensitizing a path from the fault site to a primary output [9]. The ATPG process for a targeted fault consists of two phases: fault activation and fault propagation. Fault activation establishes a signal value at the fault model site that is opposite of the value produced by the fault model. Fault propagation moves the resulting signal value, or fault effect, forward by sensitizing a path from the fault site to a primary output.

Fault Activation means to set primary input PI values such that it causes the line having the fault v to the value v' . This is an instance of the line justification problem which deals with finding an assignment of PI values those results in a desired value setting on a specified line in the circuit. Where as the term Fault Propagation

means to make the primary output bear value such as the fault is on that particular point itself. The program for ATPG was developed in C++ language. the flow chart is shown in figure 1. Here we scanned 3 netlist, one for the circuit for which test patterns are to be find, second output of testability measures sothat we can decide which input to be justify using controllability values and which path is to select for error propagation using observability values. Line justification function shown by part “J” in figure1 was developed separately. Flow chart of Justification function is shown in figure 2. The selection of any one input was done based on controllability 0 and 1 functions. The CC0 (combinational controllability 0) and CC1 (combinational controllability 1) values for both the fanins were identified. If the value to justify was 0, compared both the fanins for CC0 and selected the fan in with minimum CC0. If value to be justify was 1, compared both the fanins for CC1 and selected the fan-in with minimum CC1. Error propagation function is shown in figure 3. If given net was not a primary output but it was a stem then propagation forwarded to any of its branches. Selection of branch was on the basis of observability function. The branches of given stem and index for each branch was identified. The observability for each branch from corresponding element of obs array was identified. The branch with minimum observability was selected and the propagation function was called for selected branch with value for propagation error value (stuck value). We used the output of testability measures program to get the values of controllability 0-CC0, Controllability 1-CC1 and observability. Testability measures helped us to select a line for justification and to select a path for error propagation .Hence this ATPG was based on controllability and observability.



2.1 Genetic Approach for Test Pattern Generation:

In past, test generation using deterministic & fault oriented algorithm is highly complex and time consuming new approaches are needed to augment the existing techniques, to reduce execution time and to improve fault coverage. GA was first used for simulation based test generation in [17]. Several approaches to test generation have been proposed in [4], [9]. In reference [4], [9] the fitness evaluation and population scoring is low cost and only based on the fault coverage of each test vector. The disadvantage of the technique is that if a dropping fault simulation is used, experimentally after almost 10 generation, the generated vectors stop detecting remaining faults. This method has resulted in better final test set, but it is very expensive. A new operator is used in [4], in which, after each generation, the best vector in population is put on the final test set and then rescored with a new decreasing fitness.

2.1.1 Introduction: Genetic algorithms are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. As you can guess, genetic algorithms are inspired by Darwin's theory about evolution. Simply said, solution to a problem solved by genetic algorithms is evolved. As a result, a new random based test pattern generation technique based on GA is presented. Experimental result show that this algorithm increases fault detection rate while test size decreases. This is repeated until

some condition (for example number of populations or improvement of the best solution) is satisfied.

2.1.2 Outline of the Basic Genetic Algorithm

1. [Start] Generate random population of n chromosomes (suitable solutions for the problem) 2. [Fitness] Evaluate the fitness $f(x)$ of each chromosome x in the population [New population] Create a new population by repeating following steps until the new population is complete 1. [Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)

2. [Crossover] with a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.

3. [Mutation] with a mutation probability mutate new offspring at each locus (position in chromosome). 4. [Accepting] Place new offspring in a new population

3. [Replace] Use new generated population for a further run of algorithm

3 USING GENETIC ALGORITHM IN ATPG:

In this paper, a genetic algorithm approach to ATPG is used. The set of solutions called population is the test vectors. The purpose of this algorithm is to finding optimal solution with high convergence speed. A random population of n chromosome is generated and fitness of each chromosome in the population is evaluated. New population is created by repeating selection, crossover, mutation and acceptance.

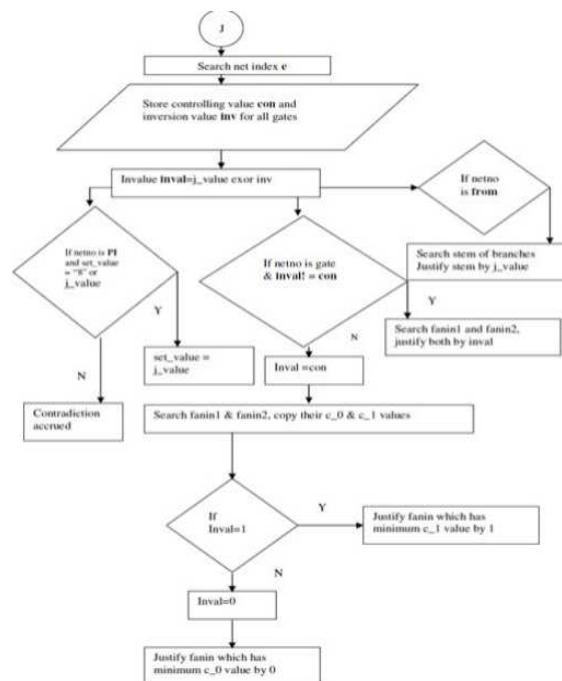


Figure 2. Flow chart of Line justification

4. TEST VECTOR COMPACTION

Amount of Data required to test ICs is growing rapidly in each new generation of technology. Increasing integration density results in larger designs with more scan cells and more faults. Moreover, achieving high-test quality in ever-smaller geometries requires more test patterns. The test vectors generated by ATPG discussed in chapter 6 and 7 can be compress. In this chapter the method to compress the test vector is discussed. As the complexity of very large scale integration (VLSI) circuit increases, testing plays an important role in today's system design. One of the most important factors in driving up the test cost is increasing the amount of test data volume, which is a result of the large size of the designs and the new types of defects appearing in the advanced manufacturing process. A large amount of test data must be stored in the automatic test equipment (ATE) and transferred deep into the chip as fast as possible. Since the channel capacity and the size of memory of ATE are limited, the test application time and the test power have been significantly increased.

a bottleneck with regards to how fast a chip can be tested. The chip cannot be tested any faster than the amount of required transferring the test data, which is, equals to: (Amount of test data on tester)/(number of tester channels x tester clock rate)

Variables and file pointers were defined. File was open in read mode “r” for which test vectors to be compressed and other for storing results in write mode “w”. while loop was used which scan the variable of file (file for which test vector is to be compress) upto the end of file .If variable c (character) is not equal to end of the line then scan each character of row one by one, else if c=end of the line then increment row and scan the character one by one upto the last column. File scanning, character by character was completed. Linematch variable was initialized to 0. x is the variable for base line. Base line is the row for which we are comparing other rows. If it is base line (take one variable i, if $i \neq x$) then for all column compare values of base line column one by one. If both the rows (baseline row and any other row) matches then discard the linedated row .Store its net no and stuck at fault. In the comparing of the column values, if baseline contains 8 (don't care X) then the other respective column should contain same value 8. In this loop we had find same test vector. We were not bother about don't care values. This was solved in next loop. Increment the row; now the base line is different do same as above. While doing so the one matched line was discarded .Then decrement the row size by 1. If base line column contains “0” or “1” and other line's corresponding column contains “8” then column value matches. Since 8 may be either “0” or “1”. The matched lines along with their faults were printed in the output file.

5. CONCLUSION

The object oriented generic program was developed to reduce the number of single stuck at faults using fault equivalence method. For

these reduced faults, test vectors were generated by controllability and observability aided ATPG based on line justification and error propagation. The test vector compaction algorithm was developed to reduce test vector data genetic algorithm solves many search and optimization problem effectively.

REFERENCES:

- [1] D. Whitley Sokolov, A. Sanyal and Y. Malaiya. Dynamic power minimization during combinational circuit test as a travelling salesman problem. In the proceeding of the IEEE Congress on Evolutionary Computation, vol.2, pp. 1088-1095, Sept 2005.
- [2] Jennifer Dworak, Michael R. Grimaila, Sooryong Lee, Brad Cobb and M. Ray Mercer. A new ATPG algorithm to limit test set size and achieve multiple detections of all faults. In the proceedings of Design, Automation and Test, Europe, 2002.
- [3] Nur A. Toubia. Survey of Test vector Compression Techniques. In the IEEE Design and Test of Computers, July-August 2006. [4] S.C. Seth, L. Pan and V.D Agrawal, PREDICT: Probabilistic Estimation of Digital Circuit Testability. In the Proceedings of Fault Tolerant Computing System, pp. 220-225, 1985.
- [5] S.C. Chang, W.B. Jone and SS. Chang. TAIR: Testability Analysis by Implication Reasoning. In the IEEE Transaction on CAD, Volume 19, No. 1, pp. 152-160, January 2000.
- [6] L.H. Goldstein, Controllability/Observability Analysis of digital circuits. In the IEEE Transactions on Circuits and Systems, Vol. 26, pp. 685-693, September 1984.
- [7] David Bryan. ISCAS'85 benchmark circuits and netlist format. North Carolina State University.
- [8] M. Abramovici and R. Aitken. Error, Fault and Defect Diagnosis: A Detective Story.



Tutorial, IEEE International Test Conference, 1997.

[9] Abramovici, Melvin A. Breuer, and Arthur D. Friedman. Digital Systems Testing and Testable Design. Jaico Publication.

[10] Michael Bushnell and Vishwani Agrawal. Essentials of Electronic Testing. Springer Publication

ABOUT THE AUTHOR:



N.Vani pursuing M.Tech (CSE) from Department of Computer Science & Engineering, Nagole institute of technology & science", Hyderabad. India. Got Distinction in "Bachelor of technology under computer science and engineering in 2013 at "Nagole institute of technology & science", Hyderabad. India.