TRANSITION-FAULT TEST GENERATION

A Senior Honors Thesis

by

BRADLEY DOUGLAS COBB

Submitted to the Office of Honors Programs
& Academic Scholarships
Texas A&M University
in partial fulfillment of the requirements of the

UNIVERSITY UNDERGRADUATE RESEARCH FELLOWS

April 2001

Group: Engineering

C

TRANSITION-FAULT TEST GEI

A Senior Honors Thesis

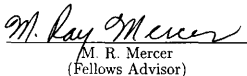
by

BRADLEY DOUGLAS COBB

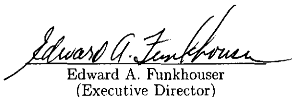
Submitted to the Office of Honors Programs
& Academic Scholarships
Texas A&M University
in partial fulfillment for the designation of

UNIVERSITY UNDERGRADUATE RESEARCH FELLOW

Approved as to style and content by:



M. R. Mercer
(Fellows Advisor)



Edward A. Funkhouser
(Executive Director)

April 2001

Group: Engineering

ABSTRACT

Transition-Fault Test Generation. (April 2001)

Bradley Douglas Cobb
Department of Electrial Engineering
Texas A&M University

Fellows Advisor: Dr. M. R. Mercer
Department of Computer Engineering

After an integrated circuit is manufactured, it must be tested to insure that it is not defective. Specifically, timing defects are becoming increasingly important to detect because of the decreasing process geometries and increasing clock rates. One way to detect these timing defects is to apply test patterns to the integrated circuit that are generated using the transition-fault model. Unfortunately, industry's current transition-fault test generation schemes produce test sets that are too large to store in the memory of the tester. The proposed methods of test generation utilize stuck-at-fault tests to create transition-fault test sets of a smaller size. Greedy algorithms are used in the generation of both the stuck-at-fault and transition-fault tests. In addition, various methods of test set compaction are explored to further reduce the size of the test sets. This research demonstrates an effective way to generate compact transition-fault test sets for a benchmark circuit and holds great promise for application to large commercial circuits.

I would like to thank Dr. M. Ray Mercer, Sooryong Lee, Dr. Mike Grimaila, and Jennifer Dworak for their contributions and support.

TABLE OF CONTENTS

CHAPTER		Page
I	INTRODUCTION	1
	A. Static and Dynamic Defects	2
	B. Stuck-at Fault and Transition Fault Models	3
	C. Excitation and Observation	4
	D. Testers and Automatic Test Pattern Generators	5
	E. Transition Fault Detection	6
II	METHOD	7
	A. Background	7
	B. Transition-Fault Test Generation Method	8
	C. Stuck-at-Fault Test Set Selection	10
	D. Compaction	10
III	RESULTS AND DATA ANALYSIS	12
	A. Transition-Fault Test Generation Method	12
	B. Stuck-at-Fault Test Set Selection	12
	C. Compaction	13
IV	CONCLUSION	16
REFERENCES		18
VITA		19

LIST OF FIGURES

FIGURE Page

1 A graphical description of an AND gate 2

2 Static and dynamic defects in an AND gate 3

3 Results of applying Method 1 and Method 2 13

4 Results of using different stuck-at-fault tests with Method 2 14

5 Results of applying reverse simulation and random simulation 14

CHAPTER I

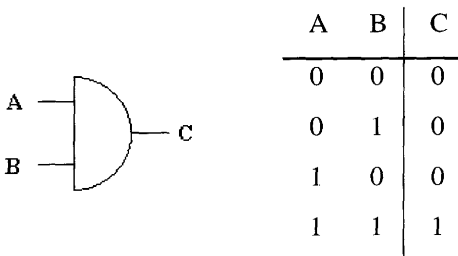
INTRODUCTION

The production of integrated circuits (IC) has recently exploded into a multi-billion dollar industry whose customers consistently demand faster and more intelligent products. In response to these demands, companies manufacture ICs that are growing increasingly larger and more complex. As with any mass produced product, a strong quality control system must be in place to assure that very few, if any, defective parts are sold. For integrated circuits, this quality control is enforced by automatic test equipment (ATE) [1]. After the IC has been produced, it is tested by the ATE to determine whether it is free from defects. The methods in which the ATE tests the circuit have been the subject of much research in the past and continue to be of great importance today.

To test an IC, the ATE enters multiple combinations of values into the circuit's inputs and observes the outputs to make sure they are correct. This is one of the only possible methods of testing because the ATE does not have access to all of the interior points in the circuit. The goal of testing is to strategically choose the inputs to the circuit so as to cause any interior defects of the circuit to manifest themselves as erroneous logic values at the circuit's outputs. One way to test for a circuit's defects is to apply every possible input combination to the circuit and verify the outputs to completely test its operation. Today's large and complex ICs cannot be tested so easily [1]. Applying every possible input combination requires 2 raised to the power of n different combinations to be applied, where n is the number of inputs to the circuit. Attempting to test in this way on a processor like an Intel Pentium III using the

---

The journal model is *IEEE Transactions on Automatic Control*.



A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Fig. 1. A graphical description of an AND gate

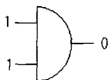
fastest ATE available today would take literally thousands of years. Clearly, a testing method that requires entering far fewer than all of the possible input combinations must be used.

To fully understand the process of testing, one must have a basic knowledge of what an integrated circuit is. An integrated circuit is made up of smaller building-block circuits called gates. The inputs to these gates can only take on the values of 0 or 1 and only produce an output of 0 or 1. The outputs of some gates are connected to the inputs of other gates in order to give the circuit a specific function. A graphical representation of one specific gate, the AND gate, is shown in Fig. 1. This figure also shows the output of the AND gate in response to all of the possible input combinations.

A. Static and Dynamic Defects

Determining which of the input combinations, or test vectors, to use depends on what type of chip defect you are targeting. There are two major categories of defects that

AND gate with a static defect



AND gate with a dynamic defect

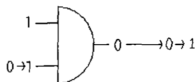


Fig. 2. Static and dynamic defects in an AND gate

can occur in a circuit: static defects and dynamic defects. Static defects occur when the circuit produces an incorrect output for a given input. Fig. 2 shows an AND gate with a static defect that generates an erroneous output of 0 when when both inputs are 1. Conversely, dynamic defects occur when the outputs of a circuit do not change quickly enough in response to changing the inputs. Fig. 2 also shows an AND gate with a dynamic defect. This defect occus when the lower input is changed from 0 to 1, causing the output of the AND gate to remain at 0 for an unacceptable amount of time before changing to the correct value of 1. Testing for static defects has been the primary concern in the past, but testing for dynamic defects is becoming increasingly important. As the clock rate of modern circuits gets faster, even tiny defects in the physical circuit can effect its timing. Therefore, both static and dynamic testing methods must be employed to effectively test state-of-the-art circuits.

B. Stuck-at Fault and Transition Fault Models

The most common way to model static defects is by using the stuck-at fault model developed by R. D. Eldred [2]. A fault model is the specification of a likely defect in the circuit. The stuck-at fault model assumes that the only defects that can occur are points in the circuit that are erroneously fixed to a 0 or a 1. This can occur

when two parts of a circuit are either erroneously connected together or not properly connected at all. For example, such a defect can force a point in the circuit to be either grounded or pulled to a high voltage regardless of what the circuit's inputs dictate it to be. Much effort has gone into the development of test vectors that can detect stuck-at faults in circuits.

Engineers have also developed ways to model dynamic defects. One of the most popular models is the transition-fault model [3]. The transition-fault model assumes that the only defects are timing delays at different points in the circuit. Specifically, transition-fault tests are written to verify that the input and output of every gate in the circuit can change from 0 to 1 and from 1 to 0 in an acceptable amount of time. Transition defects introduce unacceptable timing delays that cause the IC to fail when the inputs are rapidly switched. One of the most common causes of a transition defect is an unintended narrowing of the circuit's wiring. These nicks, caused by mishandling or manufacturing error, can affect the resistance of the wiring and in turn increase the switching time constant of a particular gate in the circuit. The delay introduced into the circuit at this point will cause the circuit's output to be invalid for an excessive period of time. While this might not present a problem for fairly static applications, delays in circuits whose inputs and outputs are rapidly changed and monitored are unacceptable.

C. Excitation and Observation

For a test vector to be successful, it must accomplish both the tasks of exciting and observing a fault. To excite a fault means to select the input vector so that it places a 0 or a 1 at a point in the circuit. For example, to excite a stuck-at-zero fault at the output of a gate in the circuit, a test vector must be generated that attempts to

place a 1 at the output of that gate. To observe a fault means to select the input vector so that it propagates the value at the desired point in the circuit to at least one of the outputs of that circuit. Extending the above example, the input vector must also be generated so that it propagates the output of the gate under test to one of the outputs. Therefore, for a test vector to detect a fault, that vector must be able to both excite and observe the fault. The process of generating vectors that detect a fault is quite tedious and is almost always done by Design Automation tools called Automatic Test Pattern Generators, or ATPGs.

D. Testers and Automatic Test Pattern Generators

Automatic Test Pattern Generators (ATPG) are computer programs designed to create a set of test vectors that completely test for faults in a circuit. Once generated, the test vectors are loaded into the memory of a tester. In order to test leading-edge circuits, testers must also be state-of-the-art. This means that the memory inside of a tester is very expensive and often smaller than optimal. As the circuits are produced, about three seconds of time in the tester is allotted to test each circuit. The limitations of tester time and tester memory create a challenge for the engineer to overcome. The test sets must conform to these limitations by having a small number of test vectors. Although most stuck-at-fault test sets will fit into a tester's memory, transition-fault test sets often will not. Also, current ATPGs are very efficient at generating tests for stuck-at faults, but have more difficulty in generating tests for transition faults.

E. Transition Fault Detection

There are three conditions that must be met in order to be able to detect a transition fault. A first test vector must be applied that sets a point in the circuit to a known value, either 0 or 1. Next, a second test vector must be applied that causes that same point to change from 0 to 1 (to detect slow-to-rise defects) or from 1 to 0 (to detect slow-to-fall defects). If a transition defect exists, this change will occur slowly when the vectors are applied to the circuit in succession, and this delay will propagate through the circuit. The final condition to meet is that this delay must propagate all the way from the point of the defect to one of the outputs where it can be observed. When choosing pairs of test vectors, the goal is to choose the smallest acceptable number of pairs that will detect all of the circuit's transition faults.

As you can see, one transition-fault test is comprised of two vectors whereas one stuck-at-fault test is comprised of only one vector. This is one of the major reasons why transition-fault test sets require more vectors, and therefore more tester memory. A second reason for this is that the average transition-fault test detects fewer faults than the average stuck-at-fault test because of the lower probability of exciting a transition fault. Because the number of stuck-at faults and transition faults is the same for a circuit (two of each at every gate input and output), it follows that more transition-fault tests will be required than stuck-at-fault tests. In order to solve this problem, a new method of automatic test pattern generation must be utilized to reduce the number of test vectors needed to test an integrated circuit for transition defects.

CHAPTER II

METHOD

A. Background

In order to experiment with different methods of transition-fault test generation, software specifications of either commercial or benchmark circuits must be used. A software specification of a circuit contains information about the kinds of gates used and how they are connected together. Because of the proprietary nature and complexity of commercial circuits, benchmark circuits are most often used in the early stages of testing research. Also, benchmark circuits provide a foundation on which to compare your research with that of others who use the same set of benchmarks. All of the experiments performed for this research were applied to a benchmark circuit known as c432 published by F. Brglez and H. Fujiwara [4]. This relatively small circuit consists of 432 sites, 864 transition faults, 36 inputs, 7 outputs, and is constructed of a variety of gates ranging from NAND gates to XOR gates.

Before performing any experiments, it is also beneficial to note the theoretical lower bound for the number of transition-fault tests needed to test c432 for all of its transition faults. A previously published work reported a lower bound of 27 tests needed to completely test c432 for stuck-at-faults [5]. This bound has been extended to transition-fault tests with one added notion. At best a transition-fault test set would use the same 27 stuck-at-fault tests as the second vectors of the transition-fault test and each second vector would be paired with a first vector that perfectly matches. By a perfect match, I mean that the first vector also excites all of the faults that are excited and observed by the second vector. While this theoretical bound is highly improbable, it nevertheless serves as a good reference point. Another valuable

reference point is the current commercial practice. The results from a simulation show that the current commercial practice detects only 322 of the total 864 transition faults.

B. Transition-Fault Test Generation Method

In this research, two distinct methods of transition-fault test generation were analyzed. Both methods involve reusing vectors that were originally generated to completely detect all of the stuck-at faults in c432. In Method 1, the stuck-at-fault test vectors are used as the only source for first and second vectors in the transition-fault test pairs. In Method 2, stuck-at-fault test vectors are still the only vectors used as second transition-fault vectors, but new first transition-fault vectors are generated. Let's first see why both Method 1 and Method 2 are guaranteed to produce a complete transition-fault test set.

The vectors of a complete stuck-at-fault test set are sufficient in themselves to be used as the only the vectors needed to create a complete transition-fault test set. To understand this, let's take a simple example related to an arbitrary point A in the circuit. By definition, there is at least one vector in the stuck-at-fault test set that sets point A to 0 and makes it observable at the outputs. Also, there is at least one vector in the stuck-at-fault test set that sets point A to 1 and makes it observable at the outputs. By making one of these stuck-at-fault vectors the first transition-fault vector and the other the second transition-fault vector, we can test either the slow-to-rise or slow-to-fall fault at that point when the vectors are applied in succession. It is quite easy to see that this process can be repeated until a test is created for both types of transition faults at every point in the circuit. Therefore, Method 1 is a valid method by which to generate a complete transition-fault test set.

Method 2 uses the same principles as stated above, but to a lesser extent. Let's take another simple example. By definition, there is at least one vector in the stuck-at-fault test set that sets point A to 1 and makes it observable at the outputs. By generating a new vector that sets point A to 0, we can apply the new vector followed by the stuck-at-fault vector to the circuit and detect if point A is slow-to-rise. In the same way, Method 2 can be repeated until a test is created for both types of transition faults at every point in the circuit. The advantage of Method 2 over Method 1 is that it has greater flexibility in selecting the first transition-fault vector. This allows for the first transition-fault vector of each test to be tailored to each second transition-fault vector.

The Method 1 tests are generated in the following manner. Each vector of the stuck-at-fault test set is paired with every other vector in the test set to find which combination results in the detection of the most transition faults. This best vector pair is then simulated on c432, and the transition faults that it detects are dropped from the objective list. This process of dropping the faults from the objective list is called fault dropping. Next, each vector is paired again with all of the others to find the combination that results in the detection of the most as-of-yet-undetected transition faults. Again, this test is simulated and fault dropping occurs. This process is repeated until a test is generated for all of the transition faults. The fewer the number of tests generated, the better the test set.

Now lets look at how the Method 2 tests are generated. First, the stuck-at-fault vector that detects the most stuck-at-faults is selected for use as the second transition-fault vector of the initial transition-fault test. A vector to be paired with the second transition-fault vector is then generated by a modified ATPG program. This transition-fault test is simulated on c432, and fault dropping removes the detected faults from the objective list. This process repeats until a test has been written

for every transition fault, each time selecting the stuck-at-fault vector that detects the most stuck-at-faults that correspond to the as-of-yet-undetected transition faults.

C. Stuck-at-Fault Test Set Selection

In addition to the two methods of transition-fault test generation, two methods of stuck-at-fault test generation were used to provide the pool of vectors for the transition-fault test generation methods. One set of stuck-at-fault tests was generated using a standard ATPG algorithm and the second set was generated using a modified version of the standard ATPG algorithm. This modified ATPG algorithm will be referred to as the greedy algorithm and was created by another member of our research group, S. Lee.

D. Compaction

Basic dynamic and static compaction methods were also applied during the experimentation of each method. Dynamic compaction refers to the process of attempting to reduce the test set size while in the process of generating vectors. Static compaction also attempts to reduce the test set size, but unlike dynamic compaction it is applied after the ATPG has generated a complete set of tests.

The three forms of static compaction used are called reverse simulation, random simulation, and test dominance. Reverse simulation involves simulating the transition-fault tests again, applying the last tests generated to the circuit first. As the simulation proceeds, the same process of fault dropping is used. In many cases, reverse simulation will show that the first tests generated are unnecessary and can be removed from the test set. Random simulation involves simulating the transition-fault test set again in a random order to detect that certain tests anywhere in the

circuit are unnecessary.

The final form of static compaction, called test dominance, involves analyzing the faults detected by each test and deterministically removing all unnecessary tests. Applying test dominance is equivalent to simulating the tests in every possible order. Although it is a bit more complicated than reverse simulation or random simulation, it guarantees that every unnecessary test will be eliminated. The same could be achieved by using random simulation indefinately, but the time needed to do this is excessive.

One form of dynamic compaction is also used. The dynamic compaction is embedded into the generation of the first transition-fault vector of every test. The algorithm that creates this vector attempts to generate it in such as way that as many transition-faults as possible are detected when it is applied with the second transition-fault vector. This form of dynamic compaction was used in all of experiments in this research.

CHAPTER III

RESULTS AND DATA ANALYSIS

A. Transition-Fault Test Generation Method

Fig. 3 shows a plot of the results from applying both methods to c432. The greedy method of stuck-at-fault test generation was used to create the stuck-at-fault vectors for this experiment.

It is interesting to note that the transition-fault tests do not detect all of the transition faults. There are some faults that, due to the nature of the circuit, cannot possibly be tested. Throughout this paper though, the phrase "all of the transition faults" is used instead of the more lengthy, though more accurate, phrase "all of the testable transition faults." For c432, there are 10 of these undetectable transition faults out of the total 864 in the circuit.

As you can see, Method 2 requires eight fewer tests than Method 1 to detect all of the transition faults. The tradeoff of using Method 2 is that its algorithm is much more complex. In general though, unless the tradeoff of complexity results in an unacceptable amount of test generation time, a complex method that produces a smaller number of tests is superior. By these criteria, Method 2 emerges as the test generation method of choice.

B. Stuck-at-Fault Test Set Selection

Having chosen Method 2 as the best transition-fault test generation method, the stuck-at-fault set must be selected that best optimizes Method 2. The results of using both a traditionally generated stuck-at-fault test set and a greedily generated suck-at-fault set are shown in Fig. 4. It is important to note that the traditional

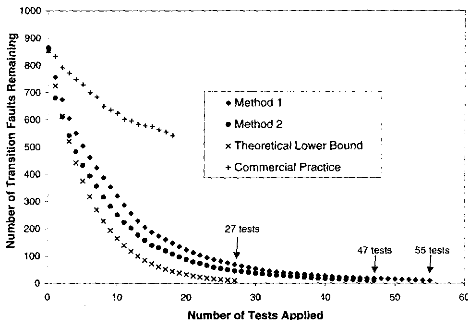


Fig. 3. Results of applying Method 1 and Method 2

stuck-at-fault test set has many more vectors (68) than the greedy stuck-at-fault test set (36). This means that the vectors in the greedy set can be thought of as being denser. The density of a vector refers to how many faults in the circuit it detects. It is obvious from these results that the greedy test set, although containing a smaller number of vectors for Method 2 to utilize, is the better of the two stuck-at-fault test sets to pair with Method 2.

C. Compaction

The next step in attempting to produce a smaller test set is to apply different forms of compaction. As mentioned earlier, both static and dynamic forms of compaction were explored. The results of applying these static compaction algorithms to Method 2 are shown in Fig. 5. The data from the random simulation is simply a representative sample from the twenty times this method was applied.

Not surprisingly, these compaction algorithms have no effect on the size of the

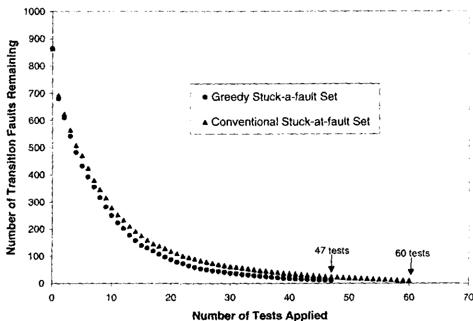


Fig. 4. Results of using different stuck-at-fault tests with Method 2

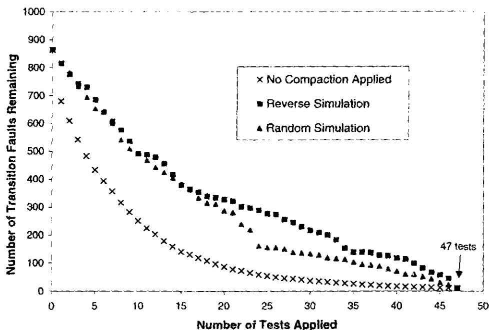


Fig. 5. Results of applying reverse simulation and random simulation

transition-fault test set (47). The reason for this is that the stuck-at-fault test set used is already highly compacted. Because the transition-fault test generation methods use these highly compacted stuck-at-fault tests, they in turn have a very low probability of further compaction. Finally, the application of test dominance to Method 2 test vectors results in the elimination of three tests, or six vectors, from the transition-fault test set (from 47 to 44). This compaction scheme is more effective than the others because it searches the test set exaustively for unnecessary tests.

CHAPTER IV

CONCLUSION

There are many ways to generate test sets for transition faults. Some methods are simple, while others are more complex. The best method is one that generates a very compact test set and does not execute too slowly when applied to large circuits. The two proposed methods of test generation both produce a complete transition-fault test set. Because of the stricter limitations on the vectors, a larger test set is generated by Method 1. By reducing the limitations on the vectors by adding customizability to the first vector of each test, Method 2 succeeds in generating a smaller, more effective test set.

Method 2 was also shown to work more effectively when given a dense set of stuck-at-fault tests from which to select the second vector of each transition-fault test. In addition, applying test dominance compaction to Method 2 succeeded in reducing the size of the test set even further. This results in a complete transition-fault test set for the benchmark circuit c432 of fourty-four tests. In comparison, the proposed methods are far superior to the current commercial practices. Though these methods were only tested on c432, their results show great potential for application to larger commercial circuits. In addition, the static compaction algorithms used in this research are predicted to be more effective when applied to test sets for larger circuits.

Future research will be carried out in two major areas. The proposed methods will be applied to a full range of benchmark circuits to gain a better understanding of their effectiveness. The proposed methods will also be extended in various ways to improve their effectiveness at detecting real defects. One approach that will be implemented consists of generating a test set that detects each transition fault mul-

tiple times instead of only once. Previous research on stuck-at faults has shown that detecting each stuck-at fault multiple times, in different ways, results in fewer defective circuits escaping undetected [6]. The same benefits will most probably arise from detecting each transition fault multiple times.

REFERENCES

[1] Jon Turino, "Semiconductor device test equipment," *VLSI Testing*, vol. 76, pp. 229–238, 1986.

[2] R. D. Eldred, "Test routines based on symbolic logic statements," *Journal ACM*, vol. 6, pp. 33–36, 1959.

[3] Barry K. Rosen John A. Waicukauski, Eric Lindenbloom and Vijay S. Iyengar, "Transition fault simulation," *IEEE Design & Test*, vol. 76, pp. 32–38, April 1987.

[4] F. Brglez and H. Fujiwara, "A neutral netlist of 10 commercial benchmark circuits and a target translator in fortran," in *Proc. IEEE Int. Symp. on Circ. Syst. (ISCAS)*, June 1985.

[5] Kozo Kinoshita Seiji Kajihara, Irith Pomeranz and Sudhakar M. Reddy, "On compacting test sets by addition and removal of test vectors," in *Proc. IEEE VLSI Test Symposium*, 1994.

[6] J. Dworak K. M. Butler B. Stewart H. Balachandran B. houchins V. Mathur J. Park L-C. Wang M. R. Grimaila, S. Lee and M. R. Mercer, "Redo–random excitation and deterministic observation–first commercial experiment," in *Proc. IEEE VLSI Test Symposium*, April 1999.

VITA

Bradley Douglas Cobb resides at 1108 Austin Avenue in College Station, TX where he is an undergraduate student in electrical engineering at Texas A&M University. His academic focuses are centered in computer engineering and include: design of digital systems, automatic test pattern generation, and communications. He is the recipient of the Engineering Scholars Program Corporate Scholarship and holds an office in Eta Kappa Nu, the electrical engineering honors society.

Previously, Mr. Cobb has worked in the areas of digital signal processing, fiber-optic sensing, and microcontroller marketing at: Texas Instruments, Stafford, TX; Input Output, Stafford, TX; and Halliburton, Houston, TX. He has one publication entitled "On the superiority of DO-RE-ME/MPG-D over stuck-at-based defective part level prediction."