

OBSERVATION-BASED TEST SET GENERATION

A Senior Honors Thesis

by

JEFFREY LEE 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 2004

Major: Electrical Engineering

OBSERVATION-BASED TEST SET GENERATION

A Senior Honors Thesis

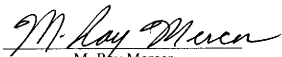
by

JEFFREY LEE COBB

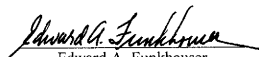
Submitted to the Office of Honors Programs
& Academic Scholarships
Texas A&M University
in partial fulfillment of the requirements of the

UNIVERSITY UNDERGRADUATE
RESEARCH FELLOWS

Approved as to style and content by:



M. Ray Mercer
(Fellows Advisor)



Edward A. Funkhouser
(Executive Director)

April 2004

Major: Electrical Engineering

ABSTRACT

Observation-Based Test Set Generation. (April 2004)

Jeffrey Lee Cobb
Department of Electrical Engineering
Texas A&M University

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

When circuits are manufactured, there are unavoidable defects that occur in a small but significant portion of the products. Input test patterns that can detect these defects are uniquely generated for each circuit in advance of their production. Current test set generation relies primarily on the “stuck-at” model, which both excites and observes every site of the circuit. However, a test set with good stuck-at fault coverage will not necessarily find all the defects in a circuit. Other models, such as bridging surrogates and transition surrogates, can also be considered when evaluating the quality of a test set. My research explores the role that observation alone plays in generating a set of valuable tests. I compare the performance of test patterns generated with traditional detection methods and ones made only considering the observation of each site. I also compare the lengths of each test set, with the goal of finding shorter and more effective tests that achieve an acceptable defective part level for a circuit.

TABLE OF CONTENTS

	Page
ABSTRACT	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	vi
LIST OF TABLES	vii
INTRODUCTION.	1
The Need for Testing.....	1
Testing Challenges	1
Modeling Faults.....	2
THE STUCK-AT FAULT MODEL	3
Detecting Stuck-At Faults	3
Shortcomings.....	5
OBSERVATION DOMINANCE	6
Excitation and Observation Comparison.....	6
Observation-Based Test Generation.....	6
ORDERED BINARY DECISION DIAGRAMS	9
OBDD Representations	9
sByDDer	10
Observation-Based Tests with sByDDer	10
RESULTS.....	12
Overview	12
Test Length.....	12
Stuck-At Fault Coverage.....	12
Bridging Fault Coverage	15
CONCLUSIONS....	18

REFERENCES.....	19
VITA	20

LIST OF FIGURES

FIGURE	Page
1 A simple circuit	3
2 AND network	7
3 (a) An AND gate (b) Boolean function (c) truth table and (d) BDD	9
4 c432 stuck-at fault coverage for n-observe tests	13
5 Observation tests compared with random tests	15
6 Bridging fault coverage comparison	16

LIST OF TABLES

TABLE	Page
1 Excitation, observation, and detection test vectors	
XYZ for circuit 1	4
2 Excitation, observation, and detection test vectors	
XY for circuit 2	7
3 Function of variable in detection test patterns	8
4 Comparison of test length	12

INTRODUCTION

The Need for Testing

Every day, companies around the world manufacture millions of computer chips and processors. As these integrated circuits come out of the assembly line, a small portion of them, usually measured in parts per million, will be defective in one way or another. These defects could be serious enough to render the entire device useless or so minor that the consumer would never notice them. Much like any other company, the manufacturers must provide some assurance that the products they sell function properly. Unfortunately, this particular verification process presents many unique challenges.

Testing Challenges

The first major challenge lies in the fact that integrated circuits are essentially black boxes with only input and output pins. Currently, no practical way exists to physically look inside a circuit and determine if it is defective. The only solution is to apply certain inputs to the circuit and verify that the outputs are correct. Ideally, the test would exercise the entire functionality of the circuit by applying every single input combination. For circuits with a small number of inputs, this method would work well, however, for modern circuits, the amount of testing required quickly gets out of hand. Consider a circuit with 40 inputs the number of unique input combinations would be 2^{40} , or more than a trillion. This number rises exponentially as the number of inputs increase. Given the limited amount of memory in the equipment used for testing, this exhaustive approach is not practical. Therefore a carefully chosen subset of all possible tests must be applied instead. Exactly how to choose these tests is where a large portion of research is directed.

This thesis follows the style and format of *IEEE Transactions on Automatic Control*.

Modeling Faults

During the manufacturing process, a number of defects could be introduced that might cause a circuit to malfunction. For example, stray metal in the circuit might cause a site to short with a ground plane. This means that the particular site would always take on the value zero, regardless of the circuit logic. Defects such as these are known as stuck-at faults, and the most widely used model for circuit defects assumes that all defects in the circuit can be traced to sites either being stuck-at-one or zero. However, other types of defects exist that the stuck-at model does not match. Transition faults occur when a site in the circuit is slow-to-rise or fall as the inputs change and bridging faults are caused by two sites being shorted together. All of these are simply ways of discretizing the infinite number of erroneous variations that could occur when manufacturing a circuit. By creating tests that attempt to detect one or more of these faults all throughout the circuit, one can determine with a certain degree of confidence whether or not a circuit is defective.

THE STUCK-AT FAULT MODEL

Detecting Stuck-At Faults

As previously stated, stuck-at faults occur when a site in the circuit takes on a value of 0 or 1 regardless of the preceding logic. In order to detect whether such a fault exists, the inputs to the circuit must be set in a way that both excites a site to a logic zero or one and allows it to be observed at an output pin. These two conditions, known as excitation and observation, are both independent of each other and necessary to determine if a stuck-at fault exists. If the value observed at the output differs from the expected value, then the circuit contains a defect.

Figure 1 shows a simple circuit made up of a single AND and OR gate. The circuit has three inputs, one output, and five individual sites with potential defects

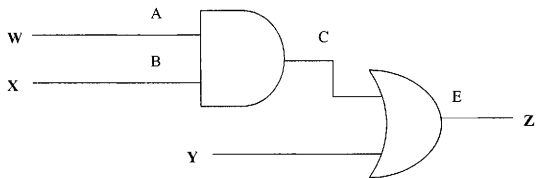


Figure 1. A simple circuit

To check for a stuck-at zero fault at site A, we set the inputs so that a logic one appears there. The actual value that appears on site A should then travel unchanged to output Z where it can be observed. To satisfy the excitation condition, input W must be a logic one, and inputs X and Y are “don’t cares” because they have no effect on exciting

site A. For the observation condition, input X is set to logic one and input Y is set to logic zero. This follows from the fact that AND gates allow signals to propagate through unchanged when the other inputs are set to one, and OR gates allow propagation when the other inputs are set to zero. The logical AND of the excitation and observation vectors is taken to find the detection test vector. Table 1 shows example excitation, observation, and detection vectors for complete stuck-at fault coverage of the circuit. Notice that each site must be observed twice, once for stuck-at zero and once for stuck-at one.

Table 1. Excitation, observation, and detection test vectors XYZ for circuit one

Site	Excitation		Observation	Detection	
	SA0	SA1		SA0	SA1
A	1DD	0DD	D10	110	010
B	D1D	D0D	1D0	110	100
C	11D	0DD	DD0	110	0D0
D	DD1	DD0	0DD	0D1	0D0
E	DD1	0D0	DDD	DD1	0D0

Although each individual test may be applied to the circuit to check for defects, this is not necessary. For example, detecting a stuck-at zero fault at site A also detects the same fault at sites B and C, since they have the same detection vectors. Along the same lines, the “don’t cares” can be set to maximize the number of other faults detected fortuitously. This means that the input assignment “010” can detect stuck-at one faults at

site A, C, D, and E. In other words, the test "010" is compatible with test "0D0," so the tests can be combined.

Shortcomings

Currently, the integrated circuit industry relies most heavily on stuck-at fault detection for determining how many actual defects a set of test patterns will uncover. Using this model will indeed find a large majority of potential defects, but prior research by Dworak in [1] has shown that this correlation does not always hold. After about 80% fault coverage, the relationship between stuck-at coverage and defect coverage declines rapidly. This means that factors other than simply exciting and observing each site of the circuit must be considered in order to create tests with better defect coverage.

OBSERVATION DOMINANCE

Excitation and Observation Comparison

When finding detection tests for a given circuit, both observation and excitation have equal importance; however, the conditions for satisfying observation are much more stringent. If we choose a random input assignment for a circuit, every site will take on either a logic one or zero. In other words, an excitation of some stuck-at fault at every site will occur regardless of whether it was specifically targeted. This same input assignment, however, is much less likely to observe many, sites in the circuit. Propagating a value in the circuit to one of the outputs requires that no other input blocks this signal path. Because of the large number of gates the signal must travel through, this usually does not happen by accident.

Along the same lines, if we consider only the excitation of every site, many defects might be uncovered at those points in the circuit. But since we can only observe the output pins, there is no way to check these internal nodes for defects. Given that a site might contain any type of defect, it must always be observed to detect its presence. It follows then that when considering only the observation of every site, a larger number of defects can be found than if only excitation were considered.

Observation-Based Test Generation

Since stuck-at fault coverage does not sufficiently test for any type of defect in the circuit, other metrics have been developed that compensate for this. For the reasons mentioned above, Dworak shows in [2] that the number of times each site is observed is one such metric. My research has focused on creating test set patterns that emphasize the importance of observation, with the goal of making more compact test sets that require less work to create.

The method that I employed to accomplish this is what I will call observation dominance. After a set of test patterns for detecting stuck-at faults is created, we can look back and determine what role each input played in either exciting or observing a

particular fault. With this information, we can compress the number of tests needed by letting inputs used for observation dominate over those used for excitation. For example, Figure 2 shows a circuit made up of a single AND gate.

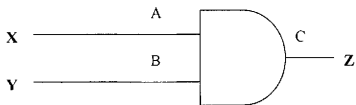


Figure 2. AND network

With two inputs, one output, and three individual sites to cover, Table 2 shows that this circuit needs very few test vectors for complete stuck-at fault coverage. Overall, there are three unique test patterns that detect all stuck-at faults in the circuit: "01," "10," and "11." The input "0D" is dominated by "01," so we remove it as redundant.

Table 2 Excitation, observation, and detection test vectors XY for circuit 2

Site	Excitation		Observation	Detection	
	SA0	SA1		SA0	SA1
A	1D	0D	D1	11	01
B	D1	D0	1D	11	10
C	11	0D	DD	11	0D

Next, we look at how each input was assigned. For example, in the test “01,” the first input came from exciting a stuck-at one at site A, and the second input came from observing site A. Table 3 shows the functions of each variable in all three test patterns.

Table 3 Function of variable in detection test patterns

Vector (XY)	Input X	Input Y
01	Excitation	Observation
10	Observation	Excitation
11	Observation	Observation

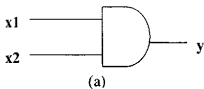
Finally, using observation dominance, we see that the first two tests are compatible with each other if the excitation inputs are regarded as “don’t cares.” The resulting test would be “11,” making that the only test needed to cover the entire circuit.

For this simple example, the number of tests was reduced from three to one, but this cut sacrificed a significant amount of stuck-at fault coverage. To see what impact observation dominance has on larger benchmark circuits, I employed the use of OBDDs to do calculations for me.

ORDERED BINARY DECISION DIAGRAMS

OBDD Representations

Binary decision diagrams (BDDs) were first introduced by Lee in [3] in the late 1950s. He created a graphical method of representing logic functions that makes them much easier to reduce and manipulate. Figure 3 shows an AND gate and three different representations of it: a Boolean function, a truth table, and a BDD.

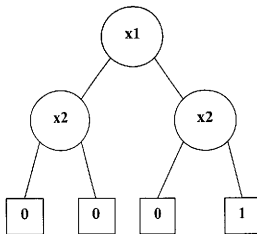


$$f(y) = x_1 * x_2$$

(b)

$x_1 x_2$	y
00	0
01	0
10	0
11	1

(c)



(d)

Figure 3. (a) An AND gate (b) Boolean function (c) truth table and (d) BDD

BDDs are simply graphical representations of truth tables. The terminal vertices, which are the squares on the bottom with either a zero or one inside, represent the output of the function depending on which branches were taken. Each variable is given a circle that branches to the left if its value is zero or to the right if its value is one. In this example, the only terminal vertex with a one comes from the path taken when both x_1 and x_2 are ones as well.

One of the main advantages of BDDs is that they are canonical forms. This means that a function has only one BDD representation, and two functions that have the same BDD representation are by definition equal. Another advantage of BDDs is that the size of their representation is almost never exponential in the number of variables, unlike Karnaugh maps which always are. This makes it much more practical when dealing with a large number of inputs like most complex circuits have. In addition, inverting functions becomes trivial, since we only have to invert the bits at the terminal vertices.

sByDDer

sByDDer is a C language program that uses OBDDs to represent logic functions. It can perform all of the basic logic operations between functions, such as AND OR and XOR, but its primary function is to create test patterns for detecting faults. In order to create stuck-at fault tests for combinational circuits, the program first creates a BDD for each site in the circuit that represents the excitation of that site. Next, it creates a separate BDD for each site that observes that point in the circuit. It then ANDs the two BDDs together to get the detection BDD for every fault in the circuit. sByDDer can also perform multiple detections at each site in order to fortuitously detect more defects in the circuit.

Observation-Based Tests with sByDDer

In order to create test sets using the principle of observation dominance, I had to modify the existing sByDDer code. The concept of treating excitation inputs as “don’t cares” and then combining compatible tests can be easily implemented using BDDs. After creating the observation BDDs, which give test patterns that observe every fault in

the circuit, the tests are taken directly from there instead of the detection BDD. The excitation BDDs are not necessary anymore, since the only condition is that every site in the circuit is observed at least once

RESULTS

Overview

To compare my results, I concentrated on a single benchmark circuit known as c432. This circuit has 432 different sites, which is small enough to make the computations quick and large enough to apply the findings to larger industry circuits. Although using a larger benchmark circuit would have given my results a larger resolution, the time needed for sByDDer to compute the BDDs grows exponentially, making it impractical for my tests

Test Length

I first looked at the reduction in test set length when only observation was considered. Table 4 below shows how many tests were generated for a given number of detections or observations per site

Table 4. Comparison of test length

n-Detect	Detection	Observation
1	31	12
2	57	18
3	73	27
4	111	37
5	138	45

On average, the observation test lengths were around 33% shorter than the detection test lengths. Clearly, much less work must be done to simply observe each site a given number of times than to excite the sites as well.

Stuck-At Fault Coverage

I then looked at what kind of stuck-at fault coverage I could achieve with the tests that only observed each site. The original detection-based tests will find 100% of

the stuck-at faults because that is what they were designed to do. My shorter observation-based tests will obviously not perform as well for this model since I did not take the excitation BDDs into consideration at all.

To test how much stuck-at coverage my tests achieved, I used a program called "Super DA". After inputting each of my test vectors for each test set, the program calculated what percentage of the stuck-at faults was detected.

Figure 4 below shows the stuck-at coverage achieved for five different test sets.

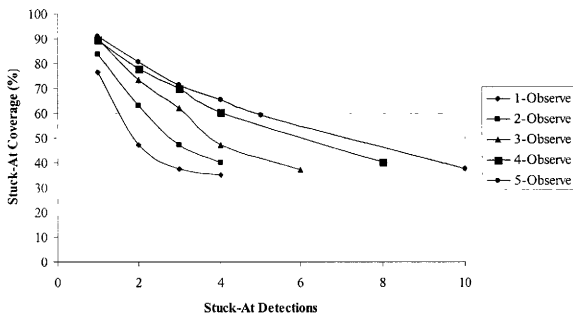


Figure 4. c432 stuck-at fault coverage for n-observe tests

The five different data series represent observation-based test sets with a different number of observations per site. The 1-observe test observes every site in the circuit once, while the 5-observe test observes each site five times.

I first looked at the effect observation alone had on stuck-at fault coverage. I compared my tests to ones that had the same number of observations per site but also

excited other sites. If we consider the 2-observe data series, we should look at the stuck-at coverage for one stuck-at detection. The reason for this is that when detecting a stuck-at fault, each site is observed twice to find both the stuck-at one and stuck-at zero faults. Since my 2-observe tests only observe each site twice, they should be compared to stuck-at coverage that observes each site the same number of times. The data shows that 84% of the stuck-at faults are detected by observing each site twice.

When looking at a larger number of observations per site, the same reasoning is used. For example, the coverage of a 4-observe data series can be directly compared to that of a 2-detect test set that both excites and observes. In this case, the data shows that 78% of the stuck-at faults were found.

Figure 4 shows that the stuck-at coverage for observation based test sets ranges from 75% to 85%. This means that the excitation condition accounts for the remaining 15% to 25% stuck-at fault coverage. I next needed to know how the observation-based tests compared to completely random test vectors. I created new tests of the same length that randomly assigned each variable to either one or zero with equal probability. Figure 5 shows how the observation-based coverage compared to that of completely random test vectors. I averaged all of the n-observe data together to get one value per stuck-at detection.

The data shows that observation alone accounts for nearly a 20% increase in stuck-at fault coverage for 1-detect when compared to a random test set. As the number of stuck-at detections increased, the difference between the two tests shortened to 6% for 4 stuck-at detections.

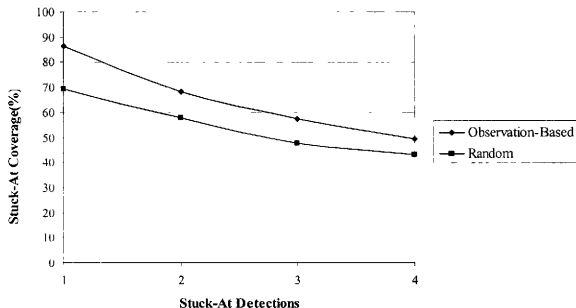


Figure 5. Observation tests compared with random tests

Bridging Fault Coverage

To compare the defect coverage of observation-based test patterns and detection-based tests, I used bridging defects as my surrogate model. The bridging surrogate model assumes that defects in the circuit come from two sites being shorted together inadvertently. I compared the defect coverage of both test sets directly since neither were created to intentionally detect bridging faults.

Since my tests were merely the stuck-at fault tests without the excitation condition, I wanted to compare the efficiency of the two. In other words, for a given test length, I wanted to find out which had the greater bridging fault coverage. Since the shortest stuck-at fault test with one detection per site was 86, I arbitrarily removed vectors from the end of the list until it matched the lengths of my observation-based tests. For control purposes, I also removed vectors from the start of the list and obtained nearly identical results. To make my observation tests match the length of the detection

tests, I increased the number of observations per site until an equal number of tests were generated.

Figure 6 below shows some of the results I achieved with test vectors created using only observation BDDs. After creating the tests, I ran them through a bridging fault simulator called "Atalanta" This program uses a given set of tests to determine what percentage of all bridging faults in the circuit was detected

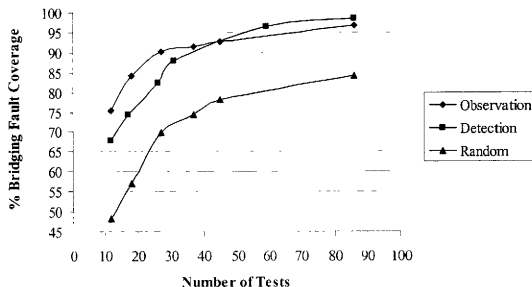


Figure 6. Bridging fault coverage comparison

The three data series represent the bridging fault coverage of three different test sets. The first set was created using only the observation BDDs, the second set was created with the detection BDDs, and the third set has random assignments for each input. All three sets have the same form, covering most of the faults with the first few tests and then requiring a larger number of tests to find the less accessible faults.

For a small number of tests, the observation-based tests found a larger percentage of the bridging faults than did the detection-based tests. With 12 tests, the difference

between the two was nearly eight percent. As the number of tests increased however, the detection tests surpassed the coverage of the observation ones around 93%. At 86 tests, the detection set had detected each site at least once, and the observation set had observed each site at least ten times, with bridging fault coverages of 99% and 97% respectively. The completely random tests quickly found 45% of the faults with 12 tests and reached its maximum of 84% coverage with 86 random vectors.

CONCLUSIONS

This research explored the role that observation vectors play in the manufacture-test generation process. To this end, an existing OBDD-based test generation tool named sByDDER was modified to generate test sets that only attempted to observe all sites in a circuit a given number of times. This was accomplished by generating tests directly from the observation OBDD instead of computing the AND of that with the excitation OBDD. The resulting test set required a significantly smaller number of tests than the detection test set for the same circuit. The observation-based tests found between 75% and 85% of the stuck-at faults that the detection-based test did with the same number of observations per site. This means that the excitation condition is necessary for only the final 15% to 25% of stuck-at fault coverage. The observation tests also found 10% to 20% more stuck-at faults than completely random tests did, which confirms the importance of observation.

When compared with the bridging surrogate, the coverage of the observation tests matched that of the detection tests very closely as the test size increased. For a small number of tests, the observation tests found more of the bridging faults than did the detection tests, but the distance closed as the number of tests grew larger. Since my observation tests required less work to create yet still nearly matched the surrogate fault coverage of the detection tests, I concluded that the observation of each site in the circuit is one of the most important factors necessary in finding circuit defects. I also found that excitation does improve defect coverage when the number of tests increases. This is because the harder to find defects require more than simply site observation to uncover.

REFERENCES

- [1] J. Dworak et al., "On the Superiority of DO-RE-ME/MPG-D Over Stuck-at-Based Defective Part Level Prediction," *Proc 2000 Asian Test Symposium*, 2000, pp. 151-157
- [2] J. Dworak et al., "Defect-Oriented Testing and Defective-Part-Level Prediction," *Design & Test of Computers*, vol. 18, no. 1, January-February 2001, pp. 31-41.
- [3] C. Y. Lee, "Representation of Switching Circuits by Binary-Decision Programs," *Bell. Syst Tech J.*, vol. 38, July 1959, pp. 985-999.

VITA

Jeffrey Lee Cobb was born in Ft. Worth, Texas on February 27, 1982. In 1993, he moved with his family to Sugar Land, Texas and attended Clements High School where he played trumpet in the marching and symphonic bands. Jeff graduated from Clements in 2000 and enrolled that year in the Electrical Engineering program at Texas A&M University in College Station, Texas. During the summers of his undergraduate career, Jeff worked at Halliburton and Dell as an engineering intern. In his senior year, Jeff participated in the Undergraduate Research Fellows Program where he studied automated test pattern generation methods under Dr. M. Ray Mercer. Jeff plans to graduate with a Bachelor of Science in Electrical Engineering from Texas A&M in December of 2004. He can be contacted at 22 Howell Lane, Sugar Land, TX 77479.