A Fault Simulation Oriented Technique for Test Point Insertion

Michael Moscovitch

McGill University, Montreal

January 1992

A Thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Engineering.

Abstract

With the ever increasing popularity of built-in self-test comes an increasing reliance on pseudo-random patterns for testing. Some faults within a circuit, however, may be hard to detect with pseudo-random patterns. One approach to improve the detectability of these faults involves modifying the circuit by adding test points so that it responds more favorably to the given test set.

A method for performing test point insertion in combinational circuits is presented. The method is based on a combination of fault simulation and probabilistic techniques.

An implementation of the test point insertion procedure is described and results are presented showing the effectiveness of this method on the ISCAS benchmark circuits.

Résumé

Avec la popularité sans cesse croissante d'autovérification pour les circuits intégrés découle une dépendance sur la qualité des vecteurs pseudo-aléatoires pour fins de test. Étant donné que certains défauts de fabrication sont difficilement détectables par des vecteurs pseudo-aléatoires, il existe des méthodes pour améliorer la couverture de ces fautes. Une de ces approches consiste à modifier le circuit par l'ajout de points de test.

Une méthode pour insérer des points de test est presentée. Elle est fondée sur deux éléments: les probabilités et la simulation de fautes.

L'implémentation de la procedure est décrite. Aussi, des résultats qui démontrent l'efficacité de cette méthode sur les circuits étalons de l'ISCAS sont présentés.

Acknowledgements

I would like to thank my thesis supervisor, Dr. Janusz Rajski for his guidance during the course of my studies. I would like to thank Fadi Maamari and Avrum Warshawski for their help with the TULIP fault simulation tool and Dimitrios Lambidonis and Eric Masson for proof reading sections of this manuscript.

I would also like to thank Jerzy Tyszer, Jacek Slaboszewicz, Michael Howells, Charles Arsenault, Rob Aitken, Henry Cox, Fidel Muradali, Ashish Pancholy and all the members of the VLSI Design Lab for their help and comments. Most of all, I would like to thank my parents for having confidence in me.

This work was supported by a scholarship from the Quebec Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR) and grants from the Natural Sciences and Engineering Research Council of Canada (NSERC).

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Improvements in technology are continually leading to increases in the level of integration possible for electronic circuits. Current commercial designs may include tens of millions of transistors on a single chip. The use of hardware description languages and synthesis tools has helped to manage the complexity in the design process.

As the complexity of VLSI circuits increase, the importance of testing procedures becomes more apparent. The marketplace not only demands more functionality but increased quality as well. To provide this quality at the system level, it is necessary to ensure that the individual components are of adequate quality.

The integrated circuit fabrication process is not perfect. Not all devices will function properly. The only way to guarantee a certain level of quality is through testing. The ideal testing procedure will identify all faulty parts.

The quality of a test can be evaluated in terms of the fault coverage that it provides. The higher the fault coverage, the better the test. Given the large circuits that may be designed today, it is no longer feasible to generate test vectors manually. Automated test pattern generation or random patterns must be used.

Unfortunately it requires more than just a desire to test chips after they are produced.

The chips themselves must be testable.

Traditionally, engineers entering the field of VLSI for the first time, were more concerned with design and function then with testing. This, of course, complicated the job of testing these integrated circuits.

Design for testability (DFT) techniques [WP83] have tried to address the issue of making chips more testable. DFT techniques such as scan design [And86, EW77, FKY89] are becoming increasingly popular. By linking all the registers of a chip into a scan chain, scan design allows a sequential circuit to be treated as a combinational one. This greatly simplifies the test generation process.

Deterministic test generation algorithms are exponential in time complexity and thus are costly in terms of cpu time. Alternatively, fault simulation may be used to select test vectors from a randomly generated sequence. Both of these methods require the storage of test patterns, typically in an external tester.

As an alternative to stored pattern testing, a pseudo random sequence may be generated on the fly within the tester or on the chip. Built-in self-test (BIST) [McC85, TS88] which integrates test vector generation and response compaction on chip reduces the need for expensive high speed external test equipment. A typical BIST implementation uses a pseudo random source, such as an linear feedback shift register (LFSR) to apply patterns to the circuit under test (CUT).

Not all circuits are susceptible to random patterns. Some faults within the circuit may be hard to detect with random patterns. One approach to improve the detectability of these faults is to adjust the test vectors using schemes such as weighted random patterns. Another approach involves modifying the circuit so that it responds more favorably to the given test set.

This thesis explores the latter approach in terms of the addition of control and observation points within the circuit. An approach to analyze the circuit and guide the placement of control and observation points is presented. Chapter 2 introduces some of the basic

issues in testing and design for testability. Chapter 3 describes the overall framework for the test point insertion method. The method itself is described in chapter 4 along with a discussion of the implementation. Results are presented in chapter 5 followed by some concluding remarks in chapter 6.

Chapter 2

Testing and Design for Testability

2.1 Fault Models

Integrated circuit fabrication is not a perfect process. The statistical nature of some of the processing steps removes any hope of obtaining a perfect device every time. Although a number of fault models exist, the stuck-at fault model [Eld59] is the most widely used. As the name implies, this model provides two failure modes for each line in the circuit. A line may be stuck-at 0 (stuck-at 1) in which case the line maintains the logic 0 (1) state in the faulty circuit.

Since each line in an n line circuit may have 3 states (fault free, stuck-at 1 or stuck-at 0), there are, 3^n combinations to consider in general. To simplify this problem, it is assumed that only one fault is present in the circuit at a time. Under this single fault assumption, there are $2n$ possible faults. Although this assumption is not necessarily true, most multiple faults are detected by tests for single faults [AF81, HM86, JB87].

Stuck at faults do not model all of the failures that can occur in VLSI circuits [BART82, PRM90]. Many other fault models have been proposed including stuck open faults [Wad78] and transition faults [Koe86, WLRI87]. Stuck-at faults, however, are commonly used be-

cause of their simplicity compared to these other models.

A full fault set for a circuit contains a stuck-at-0 and stuck-at-1 fault for each line in the circuit. Some faults in this set may be equivalent. For example a stuck-at-0 fault on the inputs of an AND gate is indistinguishable from a stuck-at-0 fault on its output. Two types of equivalence relations can be observed between faults: structural equivalence and functional equivalence [MC71]. Faults are structurally equivalent when their resulting faulty circuits have identical structures, while they are functionally equivalent when their resulting faulty circuits realize the same logic function [MC71].

Equivalent faults can be grouped into equivalence classes, and only one fault per equivalence class needs to be considered. This processes, called fault collapsing, reduces the size of the fault set without any loss of information. Most structurally equivalent faults can be identified in a linear backward pass through the circuit [SM72], such as those local to a single gate. Other equivalent faults that are harder to identify are usually ignored as the non-linear algorithm required to find them would cost more than the savings achieved through their collapsing [SM72].

A fault is untestable or redundant if its presence causes no malfunction [Fuj85]. An untestable fault results when the fault cannot be excited or the fault effect does not propagate to an observable output.

2.2 Test Pattern Generation

Integrated circuits are tested by applying stimuli to the circuit inputs and comparing the response at the outputs with the expected response. In digital circuits, the input stimuli and output responses normally consist of a sequence of binary logic values.

The combination of input and output values is known as a test pattern. The output values can be determined from the input values by logic simulation, thus the problem of test pattern generation is concerned with determining the input signal assignments.

For an n input circuit, 2^n possible input signal assignments exist. An exhaustive test, in which every possible input signal assignment is enumerated, is impractical due to time constraints.

Practical considerations dictate that the time required to generate and apply test patterns should be reasonable. There are three methods which can be used to obtain test patterns [Bot86].

1. Manual generation
2. pseudo random pattern generation
3. algorithmic (or deterministic) test generation

Manual generation is usually impractical. Pseudo random patterns are easy to generate automatically. They can be generated off chip via software methods for random number generation or on chip with an LFSR.

Deterministic test pattern generation targets specific faults within the circuit. A number of algorithms exist, including the D-algorithm, PODEM [Goe81], FAN [FS83], SOCRATES [SA88], and Quest [RC90, Cox91]. The problem of algorithmic test pattern generation is NP-complete [FT82, GJ78] so heuristics are often used to improve the performance.

Test generation for sequential circuits is an even harder problem due to the memory elements in the circuit [Kau68, BHP+71].

2.3 Testability Assessment

Given that all circuits are not always easily testable, it is necessary to have a method of assessing the testability of a circuit. In addition a suitable metric must be chosen. There are two possible types of circuit testability metrics, relative and absolute. Relative

metrics allow us to compare the testability of two or more circuits but do not give us any indication of how many faults are actually detected. A common metric of testability assessment is fault coverage. It provides an absolute measure of the percentage of faults detectable. Using fault simulation, an exact measure of fault coverage can be obtained.

2.4 Fault Simulation

Fault simulation provides a means to evaluate test quality. Given a specific set of test patterns and faults a fault simulator will determine which faults are detected by the test set. The fault coverage is then expressed as the percentage of faults detected relative to the size of the initial fault set.

Many algorithms for fault simulation have been developed. The four major categories into which most of these algorithms can be classified are those that use: fault injection methods, deductive methods, concurrent methods and critical path tracing methods [AS88].

2.4.1 Fault Injection

Fault injection is one of the most straight forward methods of fault simulation. To determine which faults are detectable by a specified test vector, a good machine simulation is performed for the test vector. The faulty circuit is simulated by injecting the fault effect at the fault site and propagating it towards the primary outputs. The fault injection process is repeated for each fault. Each fault is simulated independently of the others. The fault is considered to be detected if the fault effect reaches at least one of the primary outputs.

Once a fault has been detected, it can be removed from further simulation. This process, called fault dropping, is one of the techniques that can be used to increase the efficiency of fault simulation.

Word parallelism can also be exploited to improve the efficiency of fault simulation. During the fault simulation process, many boolean operations are performed in a repetitive manner on a large input set. Since most processors perform boolean operations in a bitwise fashion on the machine word, it is possible to use each bit to represent either a different faulty machine or a different input vector.

This technique can be applied to fault injection methods on two ways. In the parallel fault simulation method [Ses65, TS75] a computer word of w bits is associated with each line in the circuit. Each bit represents the circuit with a particular fault (or no fault). For f faults, only $\lceil f/(w-1) \rceil$ passes per pattern are required.

Alternately, w input vectors can be processed in one computer word. Thus, on a 32 bit workstation, 32 input vectors can be processed simultaneously. This exploitation of parallelism is known as the parallel pattern evaluation technique. Parallel pattern evaluation takes advantage of the full width of the processor registers to speed up the simulation process. Using this technique, a class of fault simulation methods known as parallel pattern single fault propagation [WEF+85] (PPSFP) was developed.

2.4.2 Deductive and Concurrent Methods

Deductive and concurrent fault simulation methods rely on the use of lists representing the effect of faults on each line in the circuit. This allows them to compute which faults are detected with only one pass per pattern. In a deductive fault simulator [Arm72], a list is maintained at each line of faults for which signals on the line are different from the fault free state. Fault lists are propagated by deducing the fault list on the output of a gate from the lists on its inputs using simple set operations (union and intersection).

In concurrent fault simulation [UB74] a list of faulty gates is maintained for each gate. It contains the effect of each fault on the gate if it differs from the fault free state.

2.4.3 Critical Path Tracing

One of the goals of fault simulation is to determine the detectability of faults at the primary outputs in a given test set. A number of methods have been proposed to increase the efficiency of fault simulation. These methods exploit parallelism and structural properties of the circuit.

A scheme for speeding up fault simulation by partitioning the circuit in terms of its fanout free regions (FFRs) was described by Hong [Hon78]. A fanout-free region is defined as a maximal sub-circuit containing no internal fanout stems. The output of a FFR is either a primary output or a fanout stem. The inputs to a FFR are either primary inputs or fanout branches of fanout stems. Any fault in the circuit is detected if and only if:

a) the fault is sensitized to the FFR output line and

b) the fault is propagated from the FFR output to a primary output.

In Hong's method, the faults that propagate to their FFR output are determined using a single forward pass through the circuit. Surrogate faults are injected at the fanout stems and explicitly simulated to the primary outputs.

It was shown that criticality of lines within a FFR could be determined by a single backward pass from the output of the FFR to its inputs [AMM83]. Since explicit simulation of fanout stems is $O(n^2)$ but critical path tracing is linear in complexity, a more efficient method of fault simulation was proposed in which critical path tracing within FFRs is performed before explicit simulation of stem faults [AS86]. If there are no faults critical to the fanout stem then explicit simulation is not needed. Further efficiency can be gained by stopping the propagation of surrogate faults at dominator lines [AS87].

These techniques reduced the amount of work required in the fault simulation process. The analysis of reconvergent fanout and the definition of stem regions bounded by exit lines results in a static reduction of the area of explicit simulation since it is performed only

within stem regions [MR90b]. In addition, the dropping of circuit areas from fault-free simulation provides a dynamic reduction in the fault simulation process [MR90a].

2.5 Testability Measures

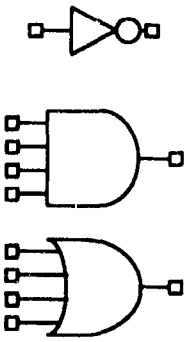
Testability measures provide information on circuit testability without the need for fault simulation. Deterministic testability measures such as the Sandia Controllability/Observability Analysis Program (SCOAP) [Gol80] are independent of the input patterns. SCOAP provides a relative measure of testability.

Probabilistic testability measures rely on primary input signal probabilities. The 0 (1) controllability of a node is defined as the probability of setting the node to a logical 0 (1) given a random input vector. Parker and McCluskey [PM75] showed and exact method of calculating the signal probabilities in a combinational circuit using exponent suppression.

It has been shown that computing the exact signal probabilities of the nodes in a general combinational circuit is NP-complete [Wun85]. Testability measures can trade off accuracy in the interest of speed. The Controllability/Observability Program (COP) [Brg84] estimates detection probabilities in linear time by not taking into account the statistical dependence of signals at a reconvergent point. The 0 and 1 controllabilities, denotes by $C_0(l)$ and $C_1(l)$ respectively for a line l, are propagated from the primary inputs to the primary outputs using the gate input/output signal probability relations described in Figure 2.1.

The Probabilistic Estimation of Digital Circuit Testability (PREDICT) [SPA85] method uses a graph approach to compute signal probabilities exactly using Shannon's expansion. An approximation procedure provides a tradeoff between accuracy and computational cost.

Statistical approaches rely on simulation data to obtain statistical estimates of controllability and observability. Statistical Fault Analysis (STAFAN) [JA84] computes signal



$$P_z = 1 - P \tag{2.1}$$

$$P_z = \prod_{i=1}^{n} P_i \tag{2.2}$$

$$P_z = 1 - \prod_{i=1}^{n} (1 - P_i) \tag{2.3}$$

Figure 2.1: Input/output signal probability relations

controllabilities from true-value simulation and then approximates observabilities deductively. The 0 and 1 controllabilities of a line l, obtained by counting the number of times l is at logic 0 or logic 1 during the true-value simulation of N vectors is:

$$C_0(l) = \frac{zero - count}{N} \tag{2.4}$$

$$C_1(l) = \frac{one - count}{N} \tag{2.5}$$

2.6 Design for Testability

To ease the test generation problem, design for testability techniques were developed. These techniques can be divided into two classes: ad hoc techniques and structured approaches [WP79].

Ad hoc techniques include circuit partitioning, and test point insertion. Structured approaches include scan design and build in logic block observation (BILBO) [KMZ79]. One of the more popular DFT techniques is scan design.

Scan design allows the registers in a sequential circuit to be connected into one or more

shift register type structures called scan chains. These scan chains can be used to serially shift data in and out of the registers thus providing observability and controllability.

Using scan design techniques, a sequential circuit can be treated as a combinational circuit for test generation purposes. This greatly simplifies the test generation problem

A natural extension of these methods is built in self test (BIST) A typical implementation of BIST uses an LFSR to generate pseudo random patterns. The output of the circuit is compacted to form a signature. A faulty signature indicates a faulty device.

2.7 Fully Testable Design

The ideal scenario for testing is to have a fully testable design. An number of methods have been proposed to accomplish this feat by modifying the circuit. Hayes showed that any circuit can be modified by embedding XOR gates so that the resulting circuit requires only five test patterns [Hay74]. The circuit is first decomposed into two input gates and all inverters are removed. XOR gates with controllable inputs and observable outputs are inserted in the input lines of every gate. A method was also given for determining the input patterns. Later in this chapter, it will be shown that XOR gates can be used to improve testability by providing a means to control the signal probabilities on specific lines in a circuit.

In general $n + 1$ test patterns are required to detect all single and multiple stuck-at faults on any n-input AND, OR, NAND, NOR gate. At least 3 test patterns are required to test a 2-input AND gate. Thus 3 can be thought of as a lower bound on the number of test patterns needed to test all stuck at faults in a circuit. Saluja and Reddy [SR74] showed that any circuit can be modified such that three test patterns are sufficient to detect all single and multiple stuck-at faults in a circuit.

The main disadvantage of these methods is that they require a large amount of circuit area. The number of gates in the circuit can easily be doubled by these methods. In

addition, the number of extra primary inputs and outputs needed can be quite large. Most practical applications cannot afford this overhead.

2.8 Testability Enhancement

With the ever increasing popularity of built in self test (BIST) and scan design, comes an increase in the reliance upon random patterns for testing. The appeal of random patterns lies in the ease with which a pseudo-random sequence can be generated on chip using an LFSR.

The effectiveness of the random patterns must be assessed. Using fault simulation to assess the coverage of randomly generated patterns we can determine if any untested faults remain. The issue of how to test the faults that are untested by the random patterns must then be addressed. Deterministic test pattern generation could be used to cover these random resistant faults, but that may require the use of two testing methods, external stored pattern testing and BIST. In fact, we may have used random patterns to avoid the use of costly deterministic test pattern generation in the first place.

BIST is one example of a methodology where a finite number of pseudo-random patterns are used to test a combinational circuit. Practical considerations such as time, limit the test length to some number of vectors N_{max}. The presence of random resistant faults, however, may require large test lengths to achieve the required coverage.

Given that the pseudo random source is in fact deterministic, such as an LFSR, the test set will always consist of the same vectors. Once a particular test set is decided upon, fault simulation can be performed on the circuit using these vectors and the coverage of single stuck-at faults can be determined. The coverage obtained, however, may not be sufficient to assure the required defect level. The problem facing us is how to improve the coverage of the given test set.

Figure 2.2 shows a typical fault coverage curve where the fault coverage is plotted as

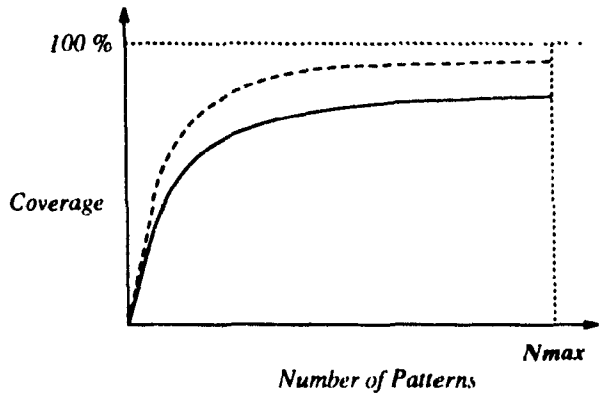


Figure 2.2: Fault coverage

a function of the number of vectors simulated. The solid curve illustrates the coverage obtained for the circuit, which may not be as high as we would like after N_{max} patterns. If we cannot afford to apply more than N_{max} vectors then we must find ways to obtain a higher coverage with those same N_{max} vectors. One way to do this is to increase the testability of the circuit and thus obtain fault coverage similar to the dashed curve. The testability can be increased with test points that allow signals internal to the circuit to be observed or controlled. Observation test points are essential if we do not want to modify the circuit but are still interested in increasing observability as in contactless probing and increased observation techniques such as cross-check [STW89]

Using scan design techniques the circuit under test can be treated as a combinational circuit. In an effort to increase the testability of these circuits and/or reduce the test lengths the use of test points has been proposed [HF74, STS91, IB89, BT86, SYKK91].

These methods rely on determining the controllability of circuit nodes and the detectability of faults through probabilistic methods and/or simulation.

2.9 Test Points

The term *test point* refers to a connection allowing a circuit node to be either observed, controlled or both. Fault detection requires that the fault be exited and the resulting fault effect propagated to an observable node in the circuit. It is not always possible to propagate the fault to existing observable nodes such as the primary outputs. Thus, it might be necessary to add a connection to some node in the circuit such that it becomes an observable output. A test point which performs this function is called an observation test point or observation point.

The function of a control test point or control point is to stimulate the propagation of faults through a gate. A control point consists of a control element which is inserted in the signal path of an input to a gate and increases the controllability of that gate input. A control point is typically used on lines that have poor controllability to adjust the signal probability to a value of 0.5. The control element consists of a gate, such as an AND, OR or XOR, of the appropriate type to affect the signal probability in the desired manner. Thus an OR gate would be used to increase the signal probability to 0.5 and an AND gate would be used to decrease it to 0.5. An XOR gate can be used to either increase or decrease the signal probability to an equiprobable state.

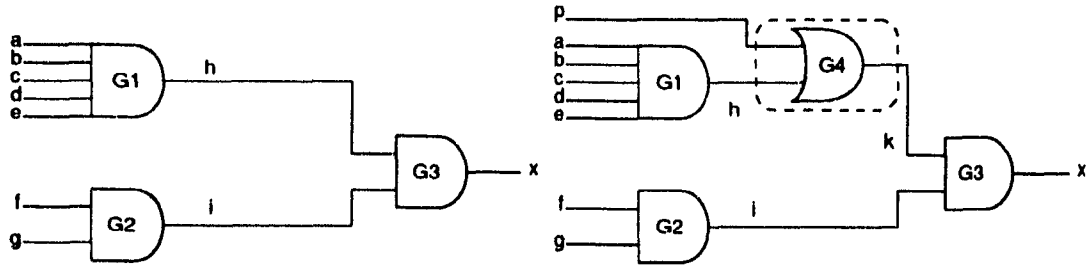


Figure 2.3: Control point

Figure 2.3a illustrates part of a circuit that contains random resistant faults. If output

h of gate $G1$ has a low signal probability or if there is some negative correlation between h and i then it may be difficult for faults to propagate through gate $G3$. To remedy this, a control point can be inserted as shown in figure 2.3b. Input p is driven by an equiprobable random source. The gate $G4$ serves to increase the signal probability of input k to gate $G3$. This will aid the propagation of faults from line i to line x.

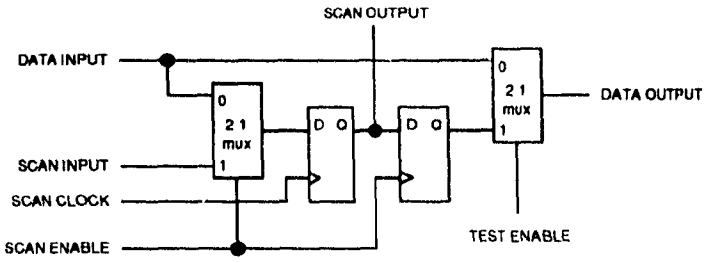


Figure 2.4: Test cell

To provide controllability and observability at a single node in the circuit, both a control point and observation point may be used. An alternative method is to use a test-cell [HHB89]. Figure 2.4 shows an implementation of a test-cell [HHB89].

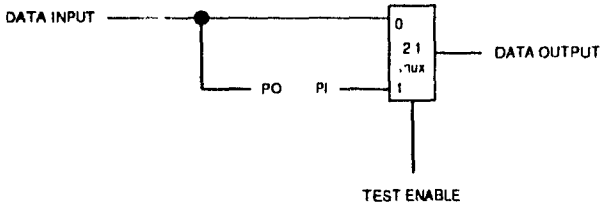


Figure 2.5: Simplified control/observe point for combinational circuit

This circuit shows some of the hardware used to construct the scan path. In the case of a combinational circuit, a more simplified method can be used as shown in Figure 2.5. The line to be observed is connected to a primary output and a multiplexor is used to select a primary input to drive the line when test mode is enabled. When test mode is disabled, the logic value on the line is unaffected. When scan design is used, the primary input and output shown in the figure will be connected to a register in the scan chain and thus the structure will be similar to Figure 2.4.

2.10 Overview of the Test Point Placement Problem

When faced with testing an integrated circuit, a test engineer's main goal is to develop a test that will minimize the number of faulty chips that are ultimately shipped to customers. In many cases this means the test should provide 100% fault coverage.

In order to improved the testability of combinational circuits, they can be modified by the addition of control and/or observation test points. The main challenge is determining the number of test points needed and their placement. The benefit of the added test points is improved coverage at the cost of increased circuit area and computation time. An optimal solution to this problem will allow complete coverage of all detectable faults with a minimum number of test points.

Considerations such as processing time may make it impractical to obtain an optimal solution for large circuits. In a real life situation, we have a limited amount of CPU time and circuit area. Since high fault coverage is our main goal, we can relax our constraints on CPU time and circuit area. Rather than minimizing them, we will require that they fall within reasonable values.

2.11 Test Point Placement

The problem of placing test points within a circuit to enhance its testability is an old one, although it is still the subject of continued research. One of the earliest methods was a labeling approach [HF74], however there were difficulties in obtaining minimal labeling for a general combinational circuit.

The placement of observation points can be determined from fault detectability information. The detectability of faults within a circuit can be determined through fault simulation or testability measures such as those described in section 2.5.

Simulation can be used to model the fault effects within a circuit. As a fault effect is

propagated through the circuit, the detectability information can be accumulated. After all the test patterns are simulated an exact measure of the detectability of each fault at each line is available. PPSFP type fault simulators have been used to guide the placement of test points [BT86, IB89].

Fault simulation can provide exact results, but has traditionally required large amounts of cpu time compared to probabilistic estimations. Probabilistic methods calculate the detectability of faults based upon some initial information such as the signal probabilities in the circuit. The signal probabilities can be determined through simulation or probabilisticly given the signal probabilities at the inputs. Probabilistic methods for computing the exact detection probability can be exponential in complexity, but, accuracy can be traded for speed.

In [STS91], the COP testability measure is use to guide the placement of test points A cost function is used to obtain a global measure for the entire circuit.

The COP testability measure is also used in [SYKK91]. Heuristics are used to solve the test point placement problem by grouping test points using the concept of fault sectors. A fault sector S is a set of faults of the same type (hard-to-control or hard-to-observe) and located in the fan-in or fanout of S. All the nodes for which the detection probability is enhanced by inserting a test point at S are grouped in a sector with its origin at S. Control (observation) points at the origin of the fault sector S cover faults in S in their fanout (fan-in) cone.

The method proposed in this thesis makes use of both fault simulation and probabilistic calculations. The techniques used are closely tied to the fault simulation framework on which the implementation is based.

Chapter 3

Framework for Test Point Placement

In this thesis, a framework for test point placement based upon fault simulation is developed. The test point placement relies on detectability information obtained from the circuit. This detectability information is obtained through simulation and probabilistic methods.

3.1 Test Patterns

Before the analysis of the circuit can begin, the test patterns and fault set must be specified. In most real life testing situations, the test patterns are deterministic. This holds for both stored patterns or pseudo random patterns such as those generated by an LFSR. Thus, the test patterns are known in advance and can be used in the analysis.

For many types of pseudo random generators such as LFSRs the patterns applied to each input is dependent on the number of inputs. It is important to maintain the same set of test patterns throughout the test point insertion process, even if primary inputs

are added to the circuit. The test patterns should be determined keeping in mind that additional primary inputs will be needed for the addition of control points.

3.2 Fault Set

The amount of memory needed for the analysis is related to the size of the fault set. It is desirable to have only those faults that we are interested in present in the fault set. Faults that are already detected can be removed from the fault set. The set of undetected faults should be used for the analysis.

This fault set is no longer valid once the circuit has been modified by the addition of control points. Control points may reduce the coverage by blocking the propagation of faults. Faults that were detected in the original circuit may be undetected in the modified circuit.

Fault collapsing can be used to reduce the size of a fault set. In a collapsed fault set, many equivalent faults can be represented by a single fault. Because of this, the coverage value obtained for a collapsed fault set may not be the same as that obtained for the full fault set. This also makes it difficult to compare the number of faults detectable by different test points. A lookup table can be used maintain the number of faults corresponding to each fault in the collapsed fault set.

3.3 Test Points

To analyze the circuit, the effect of the test points within the circuit must be measured. Rather than actually modifying the circuit under analysis, a model is used for simulation purposes. To verify the results and to create a new testable netlist a series of netlist transformations are used to insert the test points into the circuit netlist.

3.3.1 Control Elements

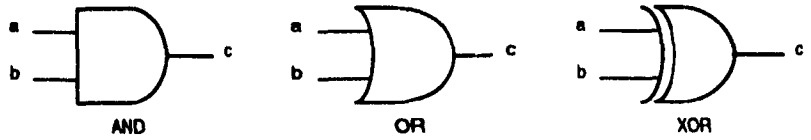


Figure 3.1: Control element types

The concept of a test point was introduced in section 2.9, where it was stated that a control point could be used to adjust the signal probability on a specific line. The three common types of control elements used for this purpose are the AND, OR and XOR gates, shown in Figure 3.1.

Each of these gates has a different effect on the signal probability of the line in which it is inserted. Using the labeling in Figure 3.1, the signal probability of the original line can be denoted by P_1^a and that of the modified line by P_1^c. The control input b will be driven by a equiprobable random source so that $P_1^b = 0.5$.

The signal probability at the output of an XOR gate is given by the following expression:

$$P_1^c = P_1^a P_0^b + P_0^a P_1^b.$$

With $P_1^b = 0.5$, this reduces to

$$P_1^c = 0.5(P_1^a + P_0^a) = 0.5.$$

The XOR gate can be used to set the signal probability to 0.5 independently of the original value. An XOR gate, however, requires more area than an AND or OR gate. The AND gate can be used to lower the signal probability according to the following relation:

$$P_1^c = P_1^a P_1^b.$$

The OR gate, which has the relation

$$P_1^c = P_1^a + P_1^b - P_1^a P_1^b,$$

can be used to raise the signal probability. Since the AND gate reduces the signal probability by 50%, the target value of 0.5 can only be reached if the original probability was 1.0. Thus it is desirable to use the AND gate only when the signal probability is close to 1. Similarly, an OR gate could be used if the signal probability is close to 0.

When control elements are inserted in a circuit, they may affect the detection probabilities of the faults that propagate through them. The effect depends on the type of control element. An XOR gate does not alter the detection probability. In the case of both AND and OR gates, the fault effect can only propagate when the other input is set to a non-controlling value. This means that the detection probability will be reduced. When a random source is used, the detection probability will be halved.

3.3.2 Simulation Models

The simulation models are used to represent the effects of a test point within the circuit. In the case of an observation point, the model is very simple. An observation point provides full observability at a particular line in the circuit but does not disturb any of the logic values within the circuit. It can be modeled by the ability to access fault detectability information at a specific line. By providing access to fault detectability information on all lines, every possible observation point can be modeled at once.

Control points require a more complicated model. They affect the logic values within the circuit. It is necessary to do some form of re-analysis for each new combination of control points. The control point can be modeled by changing the signal probability values in the circuit at the affected point and recalculating within the affected cone. For simplicity, only a single control point is modeled at a time.

The following steps are used to model a control point as shown in Figure 3.2 for simulation purposes.

- Set signal probability on line $L1$

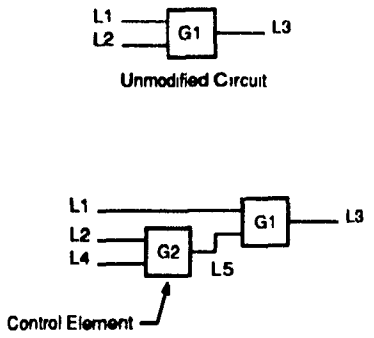


Figure 3.2: Control point simulation model

- Adjust detection probability of faults propagating through line $L1$

3.3.3 Netlist Transformations

To physically insert a test point into the circuit netlist a series of netlist transformations are defined. These local transformations insert additional lines and gates into the circuit. Each of the transformations is illustrated by showing the original section of the circuit on the left of the figure and the result of the transformation on the right. For clarity, only the gates and lines that are involved in the transformation are shown.

Observation point insertion is accomplished by the addition of a fanout stem into the netlist, in which one of the branches becomes a primary output, the other branch connects to the original destination of the line which is being made observable. There are three possible cases, which are shown in Figure 3.3.

Case 1a: The line $L1$ is an input to a stem

Add one more fanout branch to stem $S1$ ($L4$) which is a primary output

Case 1b: The line $L1$ is fanout branch of a stem

Add one more fanout to stem $S1$ ($L4$) which is a primary output

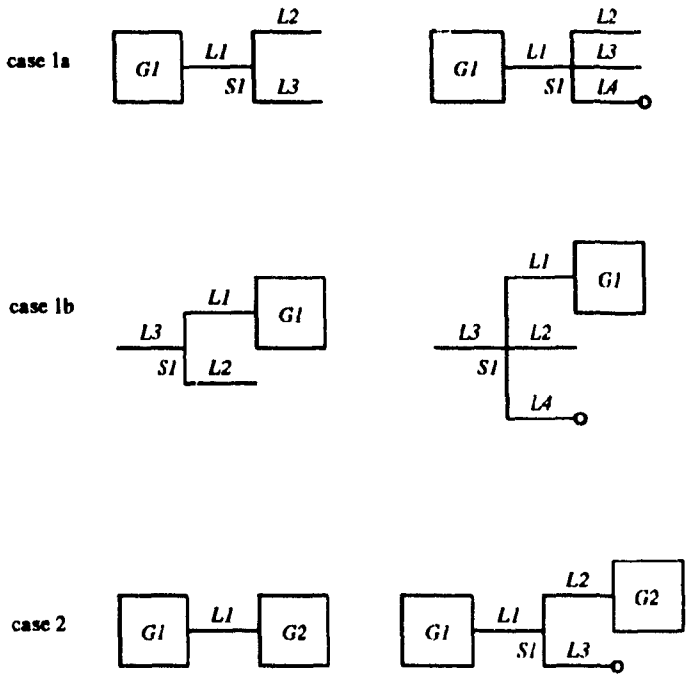


Figure 3.3: Netlist transformations for observation point addition

Case 2: The line $L1$ is between two gates, $G1$ and $G2$.

Add stem $S1$ with input $L1$ and fanout lines $L2, L3$. $L2$ connects to $G2$ in place of $L1$ and $L3$ is a primary output.

Control point insertion is accomplished by the addition of an extra controlling gate having one input controlled by a primary input of the circuit. This gate serves to modulate the signal probability of the line. There are two possible cases to consider when inserting the control point, as shown in Figure 3.4.

Case 1: The line $L1$ is an input to a gate $G1$

Add input $L3$, gate $G2$ and line $L2$.

Case 2: The line $L1$ is an input to a stem $S1$

Add input $L3$, gate $G2$ and line $L2$.

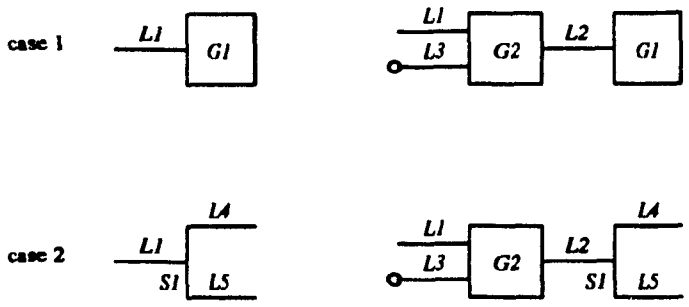


Figure 3.4: Netlist transformations for control point addition

Inserting a combined control and observation point can be done as shown in Figure 3.5. The line is split into a primary input and output. This transformation models the circuit in test mode. Under normal operation the circuit should function as it did before the transformation. This can be accomplished with a multiplexor as shown in Figure 2.5.

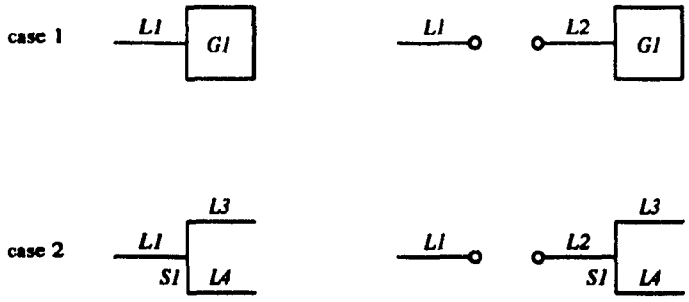


Figure 3.5: Netlist transformations for control/observe point addition

Case 1: The line $L1$ is an input to a gate $G1$

$L1$ becomes output and add new input line $L2$ which becomes input to gate $G1$.

Case 2: The line $L1$ is an input to a stem $S1$

$L1$ becomes output and add new input line $L2$ which becomes input to stem $S1$.

3.3.4 Faults in Added Hardware

Test point insertion requires the addition of circuit elements. With these additional elements come additional faults. When a circuit is modified in such a manner, its fault set should be updated to take these extra faults into account. Using the stuck-at fault model, the simplest way to accomplish this is to add a new pair of stuck-at-0 and stuck-at-1 faults on each new line added to the circuit.

3.4 Test Point Analysis Method

In order to determine the best location for test points, an analysis of the circuit is performed. This analysis consists of determining fault detectability information. The fault detectability information actually gives us information about observability in the circuit. It tells us how many faults we can expect to detect if an observation point was placed at a particular line. It does not give us information on how a control point will affect the circuit. To determine the effect of a control point, the fault detectability information must be re-evaluated in the presence of the control point. Rather than repeating a costly fault simulation, this is performed by probabilistic methods. The effect of the control points is reflected in the fault detectability information at the primary outputs of the circuit.

Fault detectability information is available for every possible observation point. Determining this information for every possible control point would be computationally expensive. To reduce the CPU time requirement, only a select set of interesting lines are evaluated as possible control points.

Chapter 4

Fault Detectability Information

This chapter presents two methods for determining fault detectability information (FDI) within a combinational circuit, the Hybrid method and the Fault Injection method. Both methods involve fault simulation. The Hybrid method is based on a combination of fault simulation and probabilistic methods. It sacrifices accuracy for speed by providing estimates of the fault detectability information. This method is used for both control and observation point placement.

The Fault Injection method is based on a parallel pattern single fault propagation (PPSFP) fault simulator. It provides exact fault detectability information. It is only used for observation point placement. As this is a simulation based method, the multiple iterations that would be required to use this method for control point placement would be costly.

4.1 The Hybrid Method for Observation Points

In this section we describe a method for estimating the number of faults detectable at a particular line in a combinational circuit. This information is then used to guide the placement of observation points. This method uses a hybrid of fault simulation and

probabilistic methods. The method consists of two phases. First, fault simulation is performed to determine controllability and fault detectability within fanout-free regions of the circuit. The fault simulation is based upon the Tulip [MR90a] simulator. The detection probabilities are then propagated from the fanout stems towards the primary outputs using probabilistic methods.

Section 4.1.1 begins with a description of the fault simulation procedure and then describes the calculation of the detection probabilities in section 4.1.2.

4.1.1 Fault Simulation

Efficient fault simulation techniques that avoid simulating parts of the circuit since they are concerned with detectability at the primary outputs, complicate the determination of detectability information within the circuit. Critical path tracing, performed within fanout-free regions from the fanout stems, does not explicitly guarantee the information on what faults are detectable and how far they propagate. The simulation of stem faults can be avoided in some cases by determining the detectability of exit lines first. Reduction of the fault-free simulation means that controllability information may not be determined in all areas of the circuit. In order to determine detectability information within FFRs, additional tracing is performed. Detectability information outside the FFRs is determined by probabilistic methods after the fault simulation is complete. As the fault simulation progresses, detected faults are dropped (fault dropping). Active FFRs are those that contain undetected faults or exit lines of an active stem. Fault-free simulation is performed on all active FFRs. Controllability information for inactive FFRs is not needed.

The fault simulation phase provides exact controllability values for the active areas of the circuit as well as detectability values within FFRs. The fault simulation is performed according to the following algorithm.

fault_simulation{

- **Perform fault-free simulation and collect controllability information**

- Trace active FFRs and simulate stem faults
- Determine detectability within active FFRs

}

Fault simulation is used to capture the properties of reconvergent fanout, which may not be adequately modeled by probabilistic methods. The fault-free simulation provides exact controllability values throughout the circuit. To capture the correlation between signals, pattern counting is performed at each gate. In order to ease the storage requirements all possible combinations of signals on multi-input gates are not considered. Instead, only $n-1$ pairs of signals are considered for an n input gate. Moreover, the pairs that are considered are those that would arise if the network was transformed into an equivalent circuit with two input gates. The fault free patterns are counted for each pair to determine four probabilities ($P_{00}, P_{01}, P_{10}, P_{11}$) as shown in Figure 4.1, where P_{xy} indicates the probability that an x will occur on the first input and a y will occur on the second.

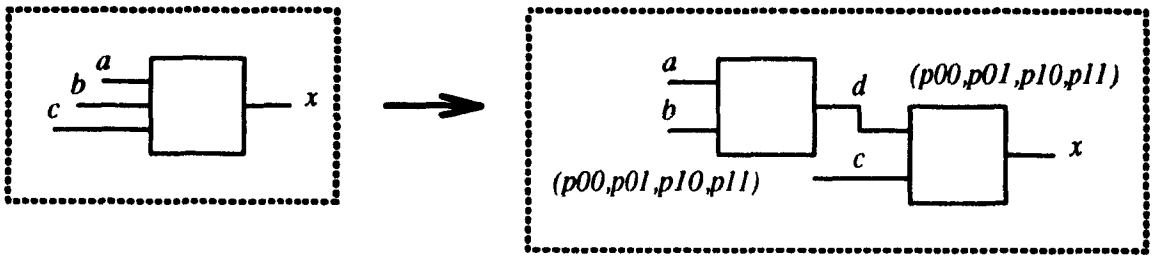


Figure 4.1: Gate input pattern counts

Within each fanout-free region detectability information is collected for those faults internal to the region. Critical path tracing is performed within fanout-free regions with respect to each line to determine a count of the number of times a fault is detectable on that line.

To increase the diagnostic resolution, the polarity of faults is also captured. The detectability information is composed of two probabilities, P_D and $P_{\bar{D}}$, representing the

probability of detecting a specific fault as a D or $\bar{D}$ on a particular line. A list is maintained on each line containing the fault identities and detection probabilities. A fault can propagate to a line in two forms, as a D or as a $\bar{D}$. The D symbol represents a 1 in the fault-free circuit and a 0 in the faulty circuit.

4.1.2 Calculation of Detection Probabilities

Initial values of the detection probability within fanout-free region are determined from simulation as the ratio (f/T) of the detection count (f) to the total number of patterns simulated (T).

Probabilistic methods are then used to propagate the detectability information forward towards the primary outputs. Each fault is propagated in a levelized fashion starting from the output stem of the fanout-free region containing the fault.

The mappings in Table 4.1 describes the equations used to propagate the detection probabilities through a gate.

AND	0	$\bar{D}$	1	D
0	0	0	0	0
$\bar{D}$	0	$\bar{D}$	$\bar{D}$	0
1	0	$\bar{D}$	1	D
D	0	0	D	D

OR	0	$\bar{D}$	1	D
0	0	$\bar{D}$	1	D
$\bar{D}$	$\bar{D}$	$\bar{D}$	1	1
1	1	1	1	1
D	D	1	1	D

XOR	0	$\bar{D}$	1	D
0	0	$\bar{D}$	1	D
D	D	0	D	1
1	1	D	0	D
D	D	1	D	0

NOT	
0	1
$\bar{D}$	D
1	0
D	$\bar{D}$

Table 4.1: Gate input/output mappings

We now derive the propagation equations for an AND gate. Consider one of the terms (T) required to produce a D for the AND gate in Table 4.1. The probability of a D on one input and a 1 on the other is indicated as $T = P(A = D \cap B = 1)$, where T is used as a shorthand notation and A and B represent the gate inputs. We can expand this to explicitly indicate the fault-free and faulty states and write $P(A = 1 \cap A' = 0 \cap B =$

$1 \cap B' = 1$). Using Bayes law, we can convert this to a conditional probability,

$$T = P(A = 1 \cap A' = 0 \mid B = 1 \cap B' = 1)P(B = 1 \cap B' = 1)$$

where the prime indicates the faulty state of the line.

During the simulation we collect information on the correlation of the fault-free signals, however, we do not have information about the correlation between faults on different inputs to a gate. At this point, we make the assumption that faults on the inputs to a gate are independent. This, of course, is not true when reconvergence is involved and will lead to errors in the estimation. Assuming faults on A and B are independent implies that

$$P(A = 1 \cap A' = 0 \mid B = 1 \cap B' = 1) = P(A = 1 \cap A' = 0 \mid B = 1 \cap B' = 0).$$

We then remove the dependence on B' and apply Bayes law to get,

$$\begin{aligned} T &= P(A = 1 \cap A' = 0 \mid B = 1)P(B = 1 \cap B' = 1) && (4.1) \\ &= \frac{P(A = 1 \cap A' = 0 \cap B = 1)P(B = 1 \cap B' = 1)}{P(B = 1)} && (4.2) \end{aligned}$$

We can apply the same method to the first term of 4.2.

$$\begin{aligned} P(A = 1 \cap A' = 0 \cap B = 1) &= P(B = 1 \mid A = 1 \cap A' = 0)P(A = 1 \cap A' = 0) \\ &= P(B = 1 \mid A = 1)P(A = 1 \mid A' = 0) \\ &= \frac{P(B = 1 \cap A = 1)P(A = 1 \cap A' = 0)}{P(A = 1)} && (4.3) \end{aligned}$$

Substituting 4.3 back into 4.2,

$$T = \frac{P(B = 1 \cap A = 1)P(B = 1 \cap B' = 1)P(A = 1 \cap A' = 0)}{P(A = 1)P(B = 1)} \qquad (4.4)$$

Given that $P(B = 1) = P(B = 1 \cap B' = 1) + P(B = 1 \cap B' = 0)$ we can substitute $P(B = 1 \cap B' = 1) = P_{01} + P_{11} - P_D^b$ in 4.4 and get

$$T = \frac{P_{11}(P_{01} + P_{11} - P_D^b)P_D^a}{(P_{10} + P_{11})(P_{01} + P_{11})}$$

Note that T is dependent on the probability that both inputs are 1, a D type fault is present on the a input and the b input has no fault effect on it. The denominator can be simplified by representing the probability of a 1 on line a as P_1^a.

Using the above method, the equations for the AND gate are defined as,

$$\begin{aligned} P_D^c &= P(A=1 \cap A'=0 \cap B=1 \cap B'=1) + P(A=1 \cap A'=1 \cap B=1 \cap B'=0) + \\ &\quad P(A=1 \cap A'=0 \cap B=1 \cap B'=0) \\ &= \frac{P_{11}P_D^a(P_1^b - P_D^b)}{P_1^a P_1^b} + \frac{P_{11}(P_1^a - P_D^a)P_D^b}{P_1^a P_1^b} + \frac{P_{11}P_D^a P_D^b}{P_1^a P_1^b} \end{aligned}$$

$$\begin{aligned} P_{\bar{D}}^c &= P(A=0 \cap A'=1 \cap B=1 \cap B'=1) + P(A=1 \cap A'=1 \cap B=0 \cap B'=1) + \\ &\quad P(A=0 \cap A'=1 \cap B=0 \cap B'=1) \\ &= \frac{P_{01}P_{\bar{D}}^a(P_1^b - P_{\bar{D}}^b)}{P_0^a P_1^b} + \frac{P_{10}(P_1^a - P_{\bar{D}}^a)P_{\bar{D}}^b}{P_1^a P_0^b} + \frac{P_{00}P_{\bar{D}}^a P_{\bar{D}}^b}{P_0^a P_0^b} \end{aligned}$$

The equations for other gate types can be derived in a similar manner. Table 4.2 shows the equations needed for all the logic gates.

Using the equations defined above to propagate the detection probabilities, a list is associated with each line. This list contains fault identities and their detection probability. In order to provide a single measure of the relative merit of all potential observation points, the lines are ranked according to the average number of faults detectable at that line. The average number of faults detectable is calculated as,

$$N_{avg} = \sum_{i \in F} 1 - (1 - P_i)^n,$$

where n is the total number of patterns and F is the set of faults in the fault list at the line. The individual probabilities $P_i = P_D + P_{\bar{D}}$ are the probabilities of detecting a fault as either D or $\bar{D}$ on the line.

AND
$P_D^c = P(A=1 \cap A'=0 \cap B=1 \cap B'=1) + P(A=1 \cap A'=1 \cap B=1 \cap B'=0) + P(A=1 \cap A'=0 \cap B=1 \cap B'=0)$
$= \frac{P_{11}P_D^a(P_1^b-P_D^b)}{P_1^aP_1^b} + \frac{P_{11}(P_1^a-P_D^a)P_D^b}{P_1^aP_1^b} + \frac{P_{11}P_D^aP_D^b}{P_1^aP_1^b}$
$P_D^c = P(A=0 \cap A'=1 \cap B=1 \cap B'=1) + P(A=1 \cap A'=1 \cap B=0 \cap B'=1) + P(A=0 \cap A'=1 \cap B=0 \cap B'=1)$
$= \frac{P_{01}P_D^a(P_1^b-P_D^b)}{P_0^aP_1^b} + \frac{P_{10}(P_1^a-P_D^a)P_D^b}{P_1^aP_0^b} + \frac{P_{00}P_D^aP_D^b}{P_0^aP_0^b}$

OR
$P_D^c = P(A=0 \cap A'=0 \cap B=1 \cap B'=0) + P(A=0 \cap A'=0 \cap B=0 \cap B'=1) + P(A=0 \cap A'=1 \cap B=0 \cap B'=0)$
$= \frac{P_{01}(P_0^a-P_D^a)P_D^b}{P_0^aP_1^b} + \frac{P_{10}P_D^a(P_0^b-P_D^b)}{P_1^aP_0^b} + \frac{P_{11}P_D^aP_D^b}{P_1^aP_1^b}$
$P_D^c = P(A=0 \cap A'=1 \cap B=0 \cap B'=0) + P(A=0 \cap A'=0 \cap B=0 \cap B'=1) + P(A=0 \cap A'=1 \cap B=0 \cap B'=1)$
$= \frac{P_{00}P_D^a(P_0^b-P_D^b)}{P_0^aP_0^b} + \frac{P_{00}(P_0^a-P_D^a)P_D^b}{P_0^aP_0^b} + \frac{P_{00}P_D^aP_D^b}{P_0^aP_0^b}$

XOR
$P_D^c = P(A=0 \cap A'=1 \cap B=1 \cap B'=1) + P(A=0 \cap A'=0 \cap B=1 \cap B'=0) + P(A=1 \cap A'=1 \cap B=0 \cap B'=1) + P(A=1 \cap B=0 \cap A'=0 \cap B'=0)$
$= \frac{P_{01}P_D^a(P_1^b-P_D^b)}{P_0^aP_1^b} + \frac{P_{01}(P_0^a-P_D^a)P_D^b}{P_0^aP_1^b} + \frac{P_{10}(P_1^a-P_D^a)P_D^b}{P_1^aP_0^b} + \frac{P_{10}P_D^a(P_0^b-P_D^b)}{P_1^aP_0^b}$
$P_D^c = P(A=0 \cap B=0 \cap A'=1 \cap B'=0) + P(A=0 \cap B=0 \cap A'=0 \cap B'=1) + P(A=1 \cap B=1 \cap A'=1 \cap B'=0) + P(A=1 \cap B=1 \cap A'=0 \cap B'=1)$
$= \frac{P_{00}P_D^a(P_0^b-P_D^b)}{P_0^aP_0^b} + \frac{P_{00}(P_0^a-P_D^a)P_D^b}{P_0^aP_0^b} + \frac{P_{11}(P_1^a-P_D^a)P_D^b}{P_1^aP_1^b} + \frac{P_{11}P_D^a(P_1^b-P_D^b)}{P_1^aP_1^b}$

NOT
$P_D^c = P_D^a$
$P_D^c = P_D^a$

Table 4.2: Detection probability equations

4.2 The Hybrid Method for Control Points

Improving the observability of a circuit is not always enough to guarantee 100% fault coverage. There may exist faults which are not exited by the particular test set. The addition of observation points will not help these faults. The problem is the controllability of nodes in the vicinity of the fault site.

This section discusses the placement of control points within the circuit to improve the controllability of specific nodes within the circuit.

The control points are used to adjust the signal probability of a specific line to a value close to 0.5. To determine which lines are good candidates for control points, a series of heuristics are applied. The process takes place in stages. First a rough selection criteria is used to create a set of lines. The lines in this set are then examined one at a time. The fault detection probabilities are recalculated based on the line having a probability of 0.5. An average value of faults detectable at the primary outputs is then determined. After all the lines are examined, the estimates of the average number of faults detectable due to each control point site is compared.

4.2.1 Line Selection Method

The initial set of lines is created based on the signal probabilities in the circuit. The signal probability information for each line is collected during the fault free simulation. Using this information, a set of lines is created. Each line is added to the set based upon the following heuristic.

- Perform levelized traversal of gates from inputs to outputs.
- Examine each gate G which has not been dropped from fault free simulation.
- For each input L_i to gate G with non-empty list of faults on the other inputs and probability of controlling value greater than or equal to a set threshold value

- add L_t to selected set
- trace backward through the circuit from line L_t and add the input line of the first stems encountered on each path to the selected set.

4.2.2 Calculation of Signal Probability Values

The signal probability values in the unmodified circuit are obtained through simulation as shown in the following equations, where P_{xy} represents the probability of obtaining the values xy on a pair of inputs to a gate.

$$\begin{aligned} P_1^a &= P_{10} + P_{11} \\ P_1^b &= P_{01} + P_{11} \\ P_0^a &= P_{00} + P_{01} \\ P_0^b &= P_{00} + P_{10} \end{aligned}$$

To determine the effect of a control point at a particular line, the signal probability of that line is set to 0.5 and the signal probabilities are recalculated in the areas of the circuit that are affected. The signal probability values for the circuit with a control point added are calculated as in the COP testability measure. These calculations do not take into account any correlation between signals in the circuit.

$$\begin{aligned} P'_{00} &= P_0'^a P_0'^b \\ P'_{01} &= P_0'^a P_1'^b \\ P'_{10} &= P_1'^a P_0'^b \\ P'_{11} &= P_1'^a P_1'^b \end{aligned}$$

4.2.3 Calculation of Detection Probabilities

The detection probabilities of faults are measured within each fanout-free region during simulation. Detection probabilities are calculated for the whole circuit using equations in Table 4.2 derived in section 4.1.2. These equations try to take into account some of the correlation between the inputs to a gate.

To determine the effect of a control point, the detection probabilities are recalculated assuming the presence of the control point. In this case, a simpler method is used. The contribution to the detection probability of a fault on line c due to a fault on line a is calculated as the probability that the fault is present at the input a to the gate and all the other inputs have non controlling values.

$$P_D^c = (\text{Prob. non controlling value}) \times P_D^a$$

The total detection probability P_D^c is calculated as the sum of contributions from the fault on each of the inputs to the gate.

4.2.4 Selection of Control Points

For each of the candidate control points, an estimate of the average number of faults detectable at the primary outputs is computed. These values are used to rank the control points in a greedy fashion. The line which detects the highest number of faults is selected first. These faults are then removed from consideration and the process is repeated by selecting the next best line. A list of control points is generated showing the average number of faults detectable and the incremental average number of faults detectable. This list can be parsed by an external procedure to select the control points that provide some incremental benefit and insert them into the circuit.

To verify the actual improvement in testability, the circuit netlists can be modified by the addition of the control points and then fault simulated. The addition of control

points is modeled by a set of netlist transformations described in Section 3.3.3.

4.3 The Fault Injection Method for Observation Points

One of the simplest simulation based methods for determining fault detectability is based upon the PPSFP fault simulation method. As each fault is propagated through the circuit, fault detectability information can be accumulated. Only one fault simulation run is necessary to obtain the fault detectability information necessary for observation point placement. Since fault simulation is usually performed once or more to evaluate the coverage of the test set, the cost of one extra fault simulation should be reasonable. The main advantage of this method is that the detectability information is exact.

4.3.1 The Simulation Algorithm

The simulation algorithm is based upon the parallel pattern single fault propagation (PPSFP) fault simulation method. The main steps in the simulation algorithm are outlined below.

1. Place the input values at the primary inputs.
2. Perform fault free simulation and count number of logical 1s on each line.
3. Perform fault simulation
   (a) Inject fault at fault site.
   (b) Propagate fault effect and update fault detectability counters on each line encountered.
   (c) If fault was detected then drop it from further simulation

In addition to the standard fault simulation, a series of fault lists and counters are maintained on each line to keep track of:

- The number of times a logical 1 was present on a particular line.
- The number of times each fault was detected on a particular line.

The fault list on each line is composed of a linked list of fault IDs and detection counts. The detection count indicates how many times the fault was detectable at that particular line. The amount of space required to store the fault lists is $O(n^2)$ in the worst case, where n is the number of lines in the circuit. To reduce the storage requirement, a fault set containing only undetected faults can be obtained from a previous fault simulation.

4.3.2 Observation Point Placement

Using fault detectability information, the placement of observation points that cover the undetected faults can be determined. A greedy algorithm is used to select each observation point in a iterative fashion. The point that covers the largest number of undetected faults is selected first. The faults that are detected by this observation point are marked. These marked faults are not considered when counting the number of faults detectable by an observation point. The process is repeated until all the undetected faults all covered, or a specified number of observation points is reached.

Once this process is complete, a list of observation points is generated. For each observation point, the number of faults detected and the incremental number of faults detected is listed. This list can be used to manually evaluate the benefit of each additional observation point. The list can be parsed by an external procedure to automatically add the observation points to the circuit netlist.

4.4 Test Point Insertion Procedure

This chapter described a set of algorithms which can be used to determine fault detectability and sites for test point insertion. This section describes how these methods can be combined to create a test point insertion procedure. The procedure consists of two main stages. In the first stage, the Hybrid method is used to determine the placement of control points. A second stage is then performed using the Fault Injection method to determine the placement of observation points. The procedure takes the following input:

Test set The test set defines the input patterns that are used to test the circuit. In this implementation, the same pseudo-random generation technique is used at each stage thus it is only necessary to define the seed value and the number of patterns.

Netlist The netlist of the circuit to be analyzed. In this implementation, a flat netlist format is used.

Fault set The fault set defines the faults in the original circuit that are to be considered in the analysis. Additional faults may be added to the fault set to account for faults in the test point hardware.

4.4.1 Choice of Fault Set

The cost of this method in terms of CPU time and memory is dependent on the size of the fault set. From the standpoint of efficiency, it is desirable to use the smallest fault set possible. A collapsed fault set would be preferable to a full fault set. A further reduction in the size of the fault set can be obtained by removing all the faults that are detected by the test set. This provides a collapsed fault set containing only undetected faults. When test points are inserted, the circuit is modified. Additional faults due to the extra hardware are easily added.

The effect of control points on any previously detected faults should be considered when more than one stage of analysis is to be performed. If in stage one some faults become undetectable due to the insertion of a control point but these faults are not present in the fault set used in stage two, these faults will not be included in the analysis and may remain undetected. If, however, they were included in the fault set, the analysis might suggest placing an observation point to detect these faults. Thus, if the test point insertion methods are applied more than once, or if the two stage procedure is used, a new fault set will have to be created for each analysis following the insertion of a control point.

In this implementation, a new fault set is generated for each analysis. The fault set is created by performing fault simulation on a full fault set. This is not the most efficient method because of the cost of the fault simulation. For the experiments described in the next chapter, however, some of the extra fault simulation steps are already needed to verify the results.

4.4.2 Stages of Analysis

There are two major stages to the test point insertion procedure, both of which comprise of analyzing the circuit and choosing the lines at which to place test points. They are similar in that they both attempt to determine the detectability of faults within the circuit. They differ, however, in the method used for the analysis. The first uses the Hybrid method which incorporates a combination of fault simulation and probabilistic methods to compute an estimate of the fault detectability. The second uses the Fault Injection method which involves a more detailed fault simulation to measure the exact fault detectability.

The test point insertion process takes place in a number of phases as shown in Figure 4.2. The procedure begins with fault simulation and control point analysis using the Hybrid method to generate a list of lines at which to place control points. Control points

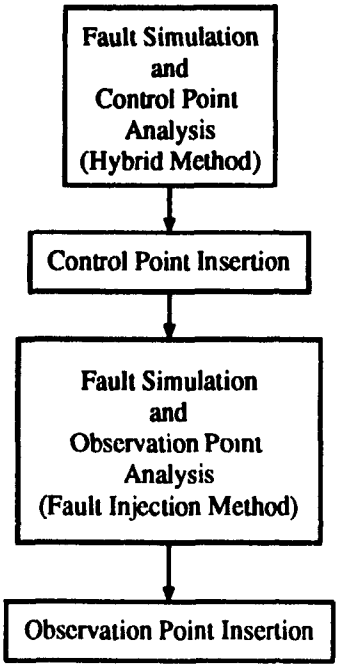


Figure 4.2: Test point insertion process

are then added to the netlist and fault simulation and observation point analysis using the Fault Injection method is performed. The observation points can then be added to the circuit.

Chapter 5

Experimental Results

5.1 Simulation Experiment Environment

To provide a basis for the CPU time measurements reported in this chapter, it is necessary to know the machine on which the experiments were performed. To maintain consistency within the results, all simulation experiments described in this chapter were performed on a SUN SPARCstation II with 32 megabytes of physical memory and 200 megabytes of swap space. This machine is rated at 25 specmarks [Sys91].

5.2 Benchmark Circuits

The experiments described in this thesis are performed on some of the ISCAS 85 [BF85] and ISCAS 89 [BBK89] benchmark circuits. Table 5.1 and 5.2 illustrate some of the properties of these circuits. The number of inputs, outputs, gates and stems is given. In addition, the number of faults in the non collapsed, collapsed and non redundant, fault sets are given. The non redundant fault set, indicated as *detectable faults*, in the table, is a collapsed fault set with the untestable faults removed. Circuits which do not

contain untestable faults or for which no list of untestable faults was available will have the detectable and collapsed faults columns equal.

Circuit Name	No. Inputs	No. Outputs	No. Gates	No. Stems	Non Collapsed Faults	Collapsed Faults	Detectable Faults
C432	36	7	160	89	864	524	520
C499	41	32	202	59	998	758	750
C880	60	26	383	125	1760	942	942
C1355	41	32	546	259	2710	1574	1566
C1908	33	25	880	385	3816	1879	1870
C2670	233	140	1193	454	5340	2747	2630
C3540	50	22	1669	579	7080	3428	3291
C5315	178	123	2307	806	10630	5350	5291
C6288	32	32	2416	1456	12576	7744	7710
C7552	207	108	3512	1300	15104	7550	7419

Table 5.1: Properties of the ISCAS 85 benchmark circuits

Fault simulation was performed on these circuits with 10240 random patterns. Tables 5.3 and 5.4 show the resulting coverage obtained. The number of undetected faults remaining, if any, is also given along with the number of patterns applied to reach the indicated coverage. For these simulations, a full fault set was used.

The coverage values shown in Tables 5.3 and 5.4 may not agree with the coverage values in other tables because of differences in the test sets. These differences may arise because Tables 5.3 and 5.4 use the original circuit descriptions for simulation. Other tables may use netlists that have been modified with extra inputs reserved for control points. These extra inputs affect the random number generation in the fault simulator and thus the test set is changed.

Circuit Name	No. Inputs	No. Outputs	No. Gates	No. Stems	Non Collapsed Faults	Collapsed Faults	Detectable Faults
C27	7	4	10	4	52	32	32
C208	19	10	96	32	416	215	215
C298	17	20	119	34	596	308	308
C344	24	26	160	40	670	342	342
C349	24	26	161	41	680	350	350
C382	24	27	158	49	764	399	399
C386	13	13	159	26	772	384	384
C420	35	18	196	66	840	430	430
C444	24	27	181	65	888	474	474
C510	25	13	211	73	1020	564	564
C526	24	27	193	54	1052	555	555
C526n	24	27	194	54	1052	553	553
C641	54	43	379	57	1278	467	467
C713	54	42	393	80	1426	581	581
C820	23	24	289	39	1640	850	850
C832	23	24	287	39	1664	870	870
C838	67	34	390	134	1676	857	857
C953	45	52	395	158	1906	1079	1079
C1196	32	32	529	155	2392	1242	1242
C1238	32	32	508	165	2476	1355	1355
C1423	91	79	657	180	2846	1515	1515
C1488	14	25	653	76	2976	1486	1486
C1494	14	25	647	76	2988	1506	1506
C5378	214	228	2779	855	10590	4603	4563
C9234	247	250	5597	1013	18468	6927	6927
C13207	700	790	7951	1224	26358	9815	9664
C15850	611	684	9772	1518	31694	11725	11336
C35932	1763	2048	16065	5295	71224	39094	35110
C38417	1664	1742	22179	4569	76678	31180	31015
C38584	1464	1730	19253	3946	76864	36303	34797

Table 5.2: Properties of the ISCAS 89 benchmark circuits

Circuit Name	Total Faults	Undetected Faults	Coverage	No. Patterns
C432	864	10	0.9884	10240
C499	998	8	0.9920	10240
C880	1760	6	0.9966	10240
C1355	2710	8	0.9970	10240
C1908	3816	12	0.9969	10240
C2670	5340	876	0.8360	10240
C3540	7080	263	0.9629	10240
C5315	10630	62	0.9942	10240
C6288	12576	68	0.9946	10240
C7552	15104	787	0.9479	10240

Table 5.3: ISCAS 85 benchmark circuits fault coverage

Some of these circuits achieve 100% coverage with the applied patterns. Since their fault coverage cannot be improved they do not provide good examples for test point insertion. These circuits will be left out of future tables.

Circuit Name	Total Faults	Undetected Faults	Coverage	No. Patterns
C27	52	0	1.0000	64
C208	416	0	1.0000	8736
C298	596	0	1.0000	384
C344	670	0	1.0000	96
C349	680	4	0.9941	10240
C382	764	0	1.0000	576
C386	772	0	1.0000	2336
C420	840	85	0.8988	10240
C444	888	22	0.9752	10240
C510	1020	0	1.0000	1184
C526	1052	2	0.9981	10240
C526n	1052	1	0.9990	10240
C641	1278	19	0.9851	10240
C713	1426	92	0.9355	10240
C820	1640	21	0.9872	10240
C832	1664	40	0.9760	10240
C838	1676	297	0.8228	10240
C953	1906	6	0.9969	10240
C1196	2392	30	0.9875	10240
C1238	2476	107	0.9568	10240
C1423	2846	35	0.9877	10240
C1488	2976	0	1.0000	5984
C1494	2988	16	0.9946	10240
C5378	10590	214	0.9798	10240
C9234	18468	2900	0.8430	10240
C13207	26358	1837	0.9303	10240
C15850	31694	2340	0.9262	10240
C35932	71224	7344	0.8969	10240
C38417	76678	4443	0.9421	10240
C38584	76864	4547	0.9408	10240

Table 5.4: ISCAS 89 benchmark circuits fault coverage

5.3 Verification of Observation Point Estimates

The Hybrid method does not produce exact results for fault detectability information. In order to verify the results, the circuit netlists are modified by placing the observation points at the selected line. The modified circuit is then fault simulated and the number of new faults detected is determined. This number is compared with the calculated value, as shown in Table 5.5.

Simulation number	Line	Calculated N_{avg}	Detected by simulation
1	1556	174.57	181
2	1559	120.00	120
3	1554	97.00	97
4	1558	95.00	95
5	1555	94.00	94
6	1557	31.00	31
7	2054	23.00	23
8	2058	23.00	23
9	2050	23.00	23
10	2062	23.00	23

Table 5.5: Number of faults detectable at observation lines in C2670

The circuits presented here part of the ISCAS 85 [BF85] and ISCAS 89 [BBK89] set of benchmark circuits. A collapsed fault set with untestable faults removed was used for these experiments. In order to simplify the comparison, the fault set was based on the faults present in the original circuit and did not take into account any new faults due to the added test points. The circuits were simulated for a maximum of 100 lines starting from the line with the highest calculated average number of faults detectable. Figures 5.1,5.2 and 5.3 illustrate the relationship between the calculated values and the values obtained from simulation. The estimates match almost perfectly for C2670 (Figure 5.1), but there

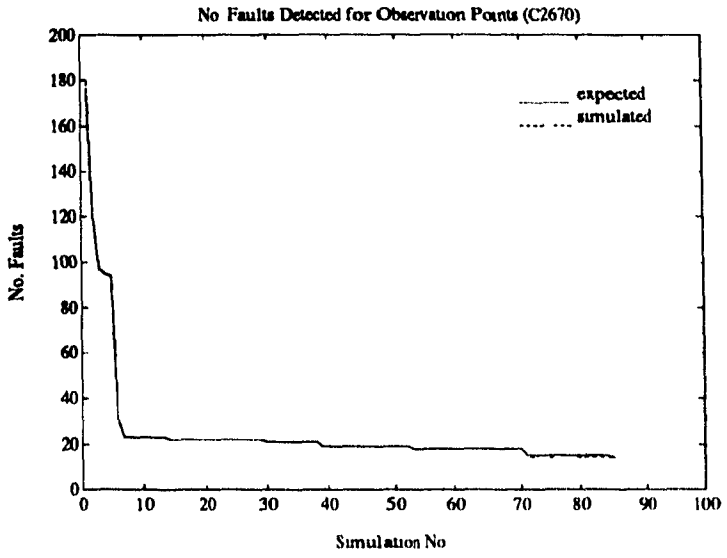


Figure 5.1: Estimated vs simulated improvement for observation points

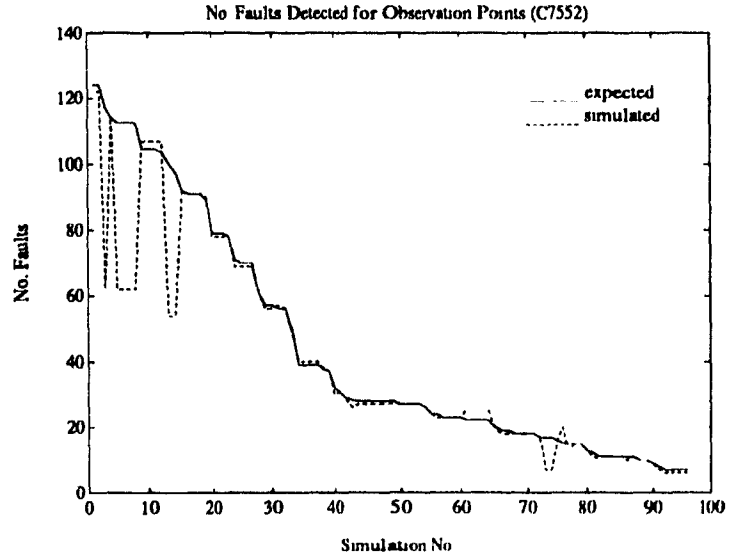


Figure 5.2: Estimated vs simulated improvement for observation points

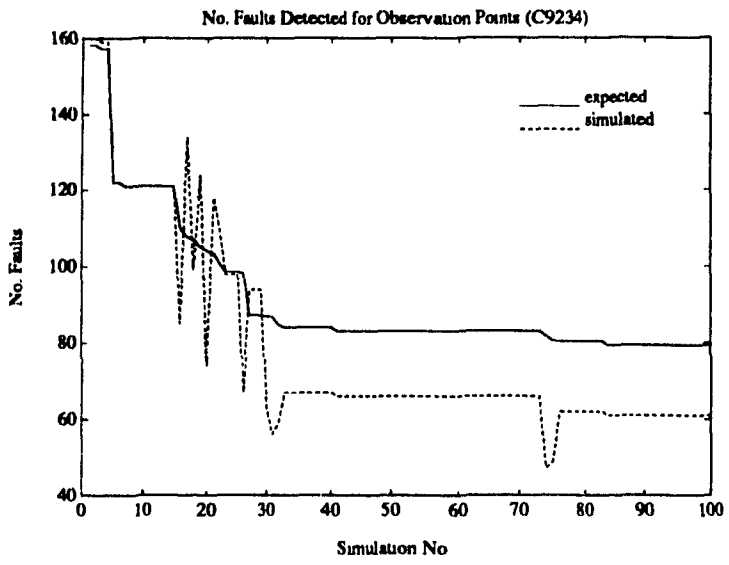


Figure 5.3: Estimated vs simulated improvement for observation points

are some differences for C7552 and C9234 (Figures 5.2 and 5.3).

While it appears that there are many lines which are overestimated for C9234 (Figure 5.3), these lines are in fact, closely related in the circuit. This area of the circuit contains groups of inverters in series and thus many lines which are equivalent in terms of the collapsed fault set.

The results in Table 5.6 show the coverage obtained after simulating 10240 random patterns with up to 3 observation points added to the circuit. The addition of observation points increased the coverage in all of the test circuits. The fault coverage curves for three of the benchmark circuits are shown in figures 5.4, 5.5 and 5.6.

Circuit	Total Faults	Fault coverage with n observation points			
		0	1	2	3
C2670	2630	88.21	95.10	99.47	99.73
C5378	4563	99.10	99.21	99.30	99.36
C7552	7419	96.00	97.64	98.41	98.49
C9234	6480	89.27	91.74	92.93	93.81
C15850	11336	94.27	95.22	96.00	96.37
C38417	31015	93.03	93.71	94.37	95.02
C38584	34797	98.71	98.76	98.80	98.83

Table 5.6: Fault coverage with varying number of observation points

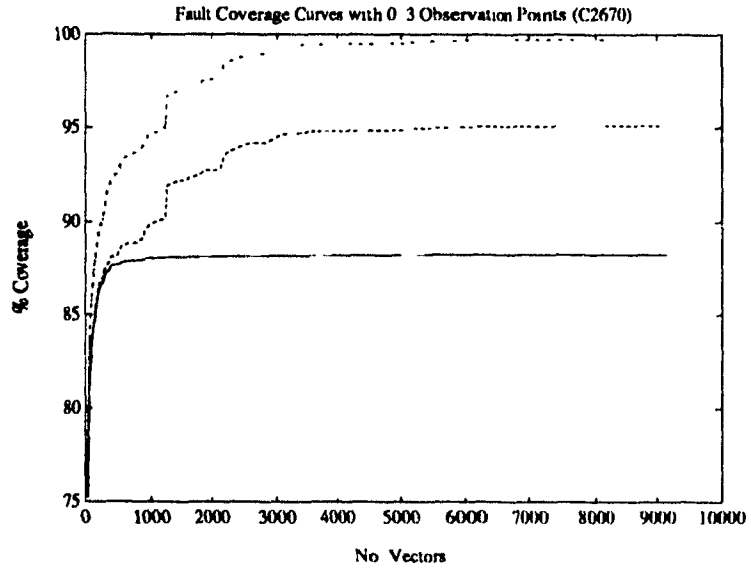


Figure 5.4: Fault coverage curves for C2670 with 0 to 3 observation points

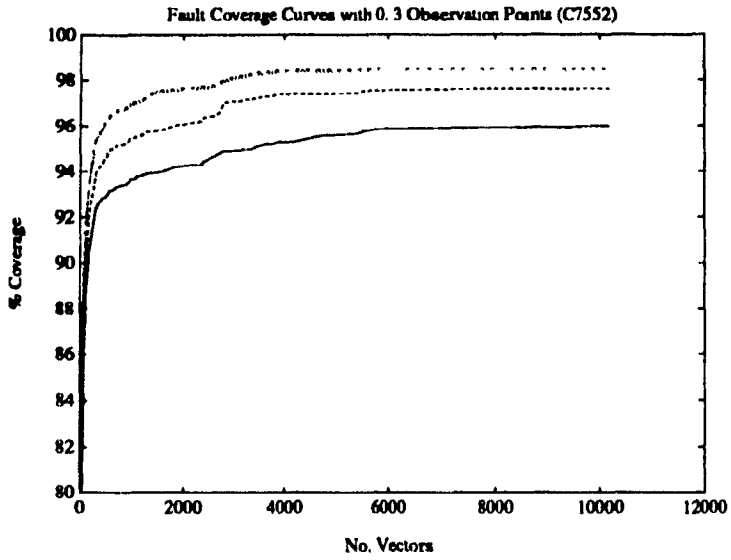


Figure 5.5: Fault coverage curves for C7552 with 0 to 3 observation points

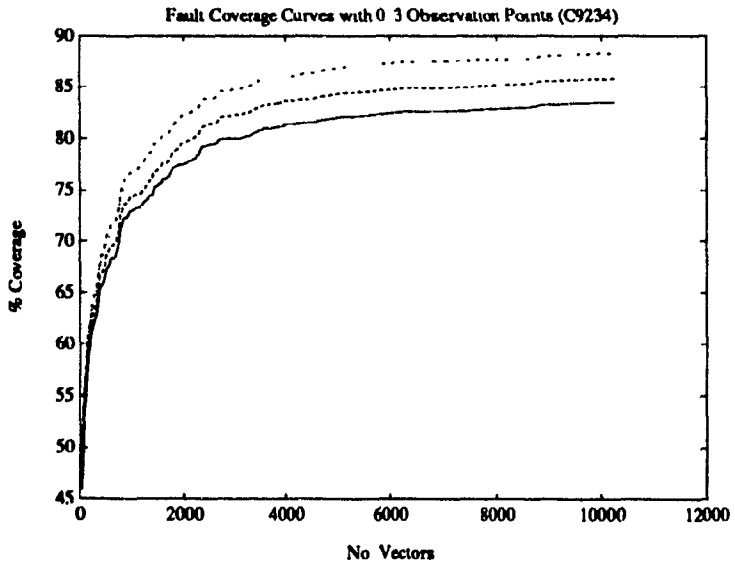


Figure 5.6: Fault coverage curves for C9234 with 0 to 3 observation points

5.4 Observation Points

The Fault Injection method provides exact results for fault detectability information. It is well suited for use as a last stage in test point insertion because the exact value for the coverage of the modified circuit can be easily determined. This removes the need to perform an extra fault simulation after the test point insertion process.

There are applications in which it is desirable to improve testability through observability only. The placement of observation points may be a useful tool for diagnosis with equipment such as E-beam testers which allow internal nodes to be monitored. In order to determine how effective observation points are at improving the testability of circuits, a series of experiments were conducted. The ISCAS benchmark circuits were analyzed by the tool and the final fault coverage, including any faults in the added hardware, was determined. Table 5.7 shows the results of this experiment. Each circuit was analyzed using both a full fault set and a collapsed fault set. The number of observation points added, the fault coverage and the CPU time required for the analysis are shown. All the fault coverage values are given in terms of a full fault set so that they can be compared. The initial coverage of the original circuit is shown only once as it is the same in both cases. This value may not match that shown in other tables because of differences in the test set.

The results obtained with both fault sets are quite similar. Circuits that reached 100% coverage, reached it with both fault sets and using the same number of observation points. Many circuits, however, did not reach 100% fault coverage. The coverage obtained with the collapsed fault set was always less than or equal to that obtained with a full fault set. When the collapsed fault set is used, there is no information on how many faults are actually detected by a specific observation point. The greedy algorithm must decide based upon the number of collapsed faults detected, thus the selection of observation points may differ from that obtained with a full fault set. For the larger circuits it is clear that more time is required when a full fault set is used.

Circuit	Full fault set				Collapsed fault set		
	Obs. pts.	Coverage before	Coverage after	cpu (s)	Obs. pts.	Coverage after	CPU (s)
C432	2	0.9940	1.0000	3	2	1.0000	5
C499	2	0.9956	1.0000	4	2	1.0000	6
C880	0	1.0000	1.0000	1	0	1.0000	3
C1355	2	0.9977	1.0000	7	2	1.0000	12
C1908	4	0.9974	0.9996	10	4	0.9996	15
C2670	51	0.8598	0.9898	110	51	0.9898	63
C3540	58	0.9662	0.9933	31	43	0.9894	36
C5315	53	0.9946	0.9996	26	53	0.9996	37
C6288	16	0.9949	0.9937	34	16	0.9937	61
C7552	75	0.9521	0.9887	119	62	0.9879	98
C349	0	0.9973	0.9973	3	0	0.9973	4
C420	9	0.9622	0.9755	5	6	0.9723	6
C444	6	0.9870	0.9965	4	6	0.9965	5
C526	1	0.9995	1.0000	4	1	1.0000	5
C526n	0	1.0000	1.0000	0	0	1.0000	2
C641	2	0.9913	0.9986	5	2	0.9986	7
C713	21	0.9591	0.9615	10	21	0.9615	10
C820	2	0.9947	1.0000	5	2	1.0000	7
C832	10	0.9858	1.0000	6	10	1.0000	8
C838	23	0.8732	0.9023	22	2	0.8791	14
C953	5	0.9915	0.9982	6	5	0.9982	9
C1196	6	0.9837	0.9978	8	4	0.9972	11
C1238	24	0.9612	0.9949	12	22	0.9943	15
C1423	16	0.9896	0.9995	10	16	0.9989	13
C1494	10	0.9958	1.0000	8	10	1.0000	11
C5378	48	0.9781	0.9938	37	36	0.9898	44
C9234	100	0.8549	0.9729	603	100	0.9651	254
C13207	100	0.9502	0.9743	157	100	0.9710	160
C15850	100	0.9304	0.9806	322	100	0.9770	247
C38417	100	0.9421	0.9909	1209	100	0.9902	707
C38584	100	0.9365	0.9585	486	100	0.9570	441

Table 5.7: Coverage before and after observation point insertion (10240 patterns)

Many of the circuits required a large number of observation points and still did not reach 100% coverage. In some of the larger circuits, the arbitrary limit of 100 observation points was reached. Observation points alone cannot solve all the testability problems in these circuits. Many of these circuits contain untestable faults. Although it is not listed in the table, it was determined during the simulation that many of the remaining faults in these circuits were not excited by the test set. Control points are needed to improve the testability of these circuits.

One of the nice features of observation points is that they do not affect the detectability of faults within the circuit except at the site of the observation point. This not only allows multiple observation points to be evaluated at the same time, but leads us to the conclusion that the fault coverage cannot be decreased by the addition of observation points. Although this is generally true there may be exceptions depending on how the faults are modeled. When the faults in the added hardware are taken into account and an observation point is added to a line which maintains a constant logic value, the number of undetected faults can be increased. This can lead to a decrease in the fault coverage. The results for circuit C6288 in Table 5.7 indicate that the fault coverage dropped from 99.49% to 99.37% after 16 observation points were added.

Experiments on circuit C6288 have shown that there are situations in which placing an observation point can decrease the coverage. The top part of Figure 5.7 shows a section of the circuit and the undetected faults in that area after fault simulation. Line 1371GAT is always 0. The bottom part of Figure 5.7 shows the circuit after an observation point is added at line 1371GAT. As this line is always 0, the s-a-0 faults on it and the observation point are not excited. Thus the modified section has one more undetected fault than the original. This phenomenon leads to a decrease in the coverage when observation points are added.

A situation such as this can be avoided by not placing observation points on lines with constant logic values such that the faults on the observation point will not be excited. When faults on the observation point cannot be excited, it is impossible to test the

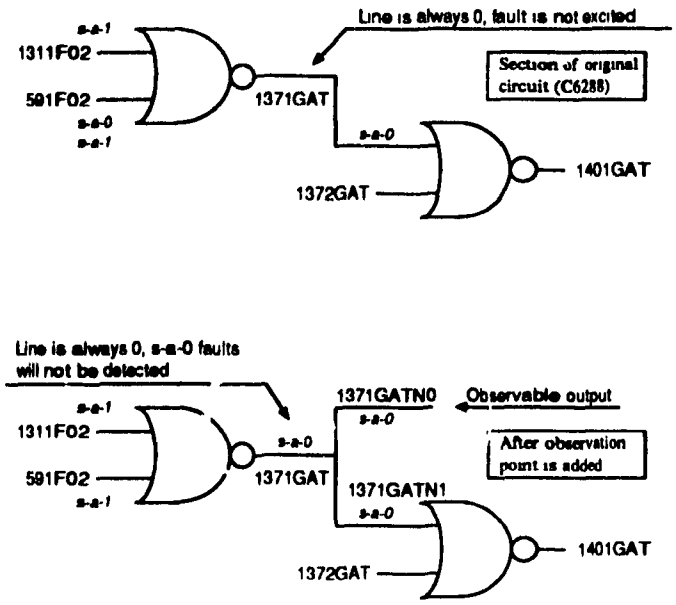


Figure 5.7: Section of circuit C6288 before and after observation point insertion

observation point itself. Control points can also be used to improve the controllability of these lines.

Table 5.7 shows the number of observation points added to the circuit, but it does not indicate how many faults are detected by each observation point. This information can be determined from the raw simulation data for each circuit. It was found that a small number of the observation points detect a large number of faults while the rest of the observation points only detect a few faults each. In fact, some observation points only detect one or two faults. Thus, if we are willing to settle for less than the maximum coverage, there is a trade off between the overhead of each additional observation point and the incremental improvement in fault coverage. Since a greedy algorithm is used, the incremental number of faults detected for each successive observation point is non-increasing.

A threshold value can be introduced to filter out all observation points that detect less faults than the threshold value. Table 5.8 shows the results with a threshold value of two

Circuit	Observation points	Coverage before	Coverage after	CPU (s)
C432	1	0.9940	0.9994	3
C499	2	0.9956	1.0000	4
C880	0	1.0000	1.0000	1
C1355	2	0.9977	1.0000	7
C1908	1	0.9974	0.9989	10
C2670	20	0.8598	0.9858	109
C3540	25	0.9662	0.9891	30
C5315	5	0.9946	0.9955	25
C6288	0	0.9949	0.9949	33
C7552	16	0.9521	0.9848	118
C349	0	0.9973	0.9973	3
C420	6	0.9622	0.9747	5
C444	3	0.9870	0.9959	4
C526	0	0.9995	0.9995	3
C526n	0	1.0000	1.0000	0
C641	2	0.9913	0.9986	5
C713	2	0.9591	0.9660	10
C820	1	0.9947	0.9996	5
C832	3	0.9858	0.9972	5
C838	22	0.8732	0.9018	22
C953	3	0.9915	0.9974	6
C1196	4	0.9837	0.9972	8
C1238	17	0.9612	0.9928	12
C1423	11	0.9896	0.9981	10
C1494	3	0.9958	0.9982	8
C5378	34	0.9781	0.9925	36
C9234	100	0.8549	0.9729	603
C13207	100	0.9502	0.9743	157
C15850	100	0.9304	0.9806	322
C38417	100	0.9421	0.9909	1198
C38584	100	0.9365	0.9585	480

Table 5.8: Observation point insertion with threshold of 2 faults

and a full fault set. Only observation points that detect two or more additional faults are included.

Some of the circuits, such as C432 and C526 no longer reach 100% coverage. The coverage for many other circuits is also slightly decreased. The number of observation points, however, is greatly decreased for some circuits like C2670 and C7552. Thus it is sometimes possible to achieve the benefit of improved testability with very few observation points.

It is also interesting to note that no observation points are added to C6288. The observation points that were added in the previous analysis only detected one additional fault each, which was negated by the additional faults added by the observation point.

5.5 Test Point Insertion

This section describes the results obtained by applying the test point insertion method described in section 4.4. To determine the actual coverage achieved, the test points are added to the netlist and fault simulation is performed. The test points are modeled using the netlist transformations described in section 3.3.3. The ISCAS 85 and ISCAS 89 benchmark circuits, described earlier in this chapter, were used for the experiments. All the fault simulation steps in a set of results used the same number of patterns.

The number of test points needed is broken down into three values. The number of control only and observe only test points are indicated by the "obs" and "ctl" columns respectively. The "co" column indicates the number of combined control and observation test points. The sum of these three columns indicates the total number of test points.

All the results were generated using a test length of 10240 patterns unless otherwise indicated. Some of the circuits, such as C880, are fully tested by the random patterns without any additional test points. These circuits have a coverage of 1.0 in both the before and after columns.

Circuit	Test Points			Coverage		CPU
	obs	ctl	co	before	after	(s)
C432	2	0	0	0.9940	1.0000	9
C499	2	0	0	0.9956	1.0000	12
C880	0	0	0	1.0000	1.0000	2
C1355	2	1	0	0.9977	1.0000	27
C1908	4	1	0	0.9974	0.9996	25
C2670	50	5	1	0.8598	0.9945	169
C3540	56	11	0	0.9662	0.9933	112
C5315	53	0	0	0.9946	0.9996	82
C6288	16	0	0	0.9949	0.9937	156
C7552	69	9	2	0.9521	0.9960	348
C349	0	0	0	0.9973	0.9973	6
C420	1	1	1	0.9622	1.0000	9
C444	4	2	0	0.9870	0.9965	8
C526	1	0	0	0.9995	1.0000	7
C526n	0	0	0	1.0000	1.0000	1
C641	0	0	1	0.9913	1 0000	6
C713	19	2	1	0.9591	0.9674	20
C820	1	0	1	0.9947	1.0000	12
C832	9	0	1	0.9858	1.0000	18
C838	16	16	1	0.8732	0.9882	42
C953	4	2	0	0.9915	1.0000	14
C1196	3	1	0	0.9837	0.9997	19
C1238	22	3	0	0.9612	0.9979	32
C1423	12	2	1	0.9896	1.0000	26
C1494	10	0	0	0.9958	1.0000	34
C5378	28	4	0	0.9781	0.9979	106
C9234	100	22	0	0.8549	0.9879	2893
C13207	100	20	0	0.9502	0.9983	4838

Table 5.9: Coverage before and after test point insertion (10240 patterns)

Table 5.9 shows the results of the test point insertion procedure using a full fault set. Two of the larger circuits were not simulated because of the memory requirements for the control point analysis phase.

In the control point analysis phase, the detectability of each fault is calculated from the detection probabilities measured during the fault simulation. Because of the probabilistic treatment of the detection probabilities, a non-zero detection probability may be calculated when in fact the fault effect has died. The estimated detection probabilities may even reach the primary outputs even though the fault is undetectable. This leads to larger fault lists than necessary and thus increased memory usage.

Not all of the circuits benefit from the test point analysis. For some of the circuits, no test points can be suggested. Circuit C349 in Table 5.9 is an example of a circuit for which the method provides no additional test points, even though the circuit is not fully tested with the applied patterns.

For those circuits requiring only observation points, the results are the same as in Table 5.7. C1432 and C641 achieve 100% coverage after test points are added.

Table 5.10 shows the results of repeating the experiments with a collapsed fault set. The coverage values, however, are given in terms of a full fault set for comparison. The reduction in the size of the fault set allowed all the benchmark circuits to be simulated. The coverage values and number of test points were slightly changed from the previous results, however, the same circuits achieved 100% coverage in both experiments. The major difference is in the CPU time required for the experiment. The smaller fault set greatly reduced the simulation time.

Table 5.11 shows the effect of increasing the test length to 102400 patterns. As expected, the initial coverage is increased, which in turn increases the final coverage. A comparison of Table 5.9 with Table 5.11 shows that in many cases a slightly higher coverage is reached with fewer test points needed. The improvement is not very large and may not be worth the extra fault simulation time. When the number of undetected faults is

	Test Points			Coverage		CPU
Circuit	obs	ctl	co	before	after	(s)
C432	2	0	0	0.9940	1.0000	13
C499	2	0	0	0.9956	1.0000	16
C880	0	0	0	1.0000	1.0000	7
C1355	2	1	0	0.9977	1.0000	36
C1908	4	1	0	0.9974	0.9996	36
C2670	50	4	1	0.8598	0.9912	114
C3540	42	3	0	0.9662	0.9896	129
C5315	53	0	0	0.9946	0.9996	105
C6288	16	0	0	0.9919	0.9937	215
C7552	59	7	1	0.9521	0.9942	281
C349	0	0	0	0.9973	0.9973	8
C420	1	1	0	0.9622	1.0000	12
C444	5	1	0	0.9870	0.9965	12
C526	1	0	0	0.9995	1.0000	10
C526n	0	0	0	1.0000	1.0000	4
C641	0	0	1	0.9913	1.0000	9
C713	19	1	1	0.9591	0.9657	23
C820	2	1	0	0.9947	1.0000	17
C832	10	1	0	0.9858	1.0000	23
C838	5	5	1	0.8732	0.9739	30
C953	3	2	0	0.9915	1.0000	20
C1196	2	1	0	0.9837	0.9994	25
C1238	21	3	0	0.9612	0.9976	38
C1423	14	2	0	0.9896	1.0000	33
C1494	10	0	0	0.9958	1.0000	32
C5378	21	2	0	0.9781	0.9943	113
C9234	100	21	0	0.8549	0.9818	768
C13207	98	16	0	0.9502	0.9970	2950
C15850	100	27	1	0.9304	0.9910	79521
C38417	100	40	1	0.9421	0.9967	35528
C38584	100	52	4	0.9365	0.9750	1831

Table 5.10: Test point insertion using collapsed fault set

small, the overhead of the extra fault simulation exceeds that of the control point analysis. An increase in the test length can improve the simulation time if it significantly reduces the number of undetected faults and thus decreases the control point analysis time. If the test length is to short, then there will be an excessive number of undetected faults. This will increase the cost of the control point analysis in terms of memory usage and CPU time.

As the fault coverage curve begins to level off, the benefit of increasing the test pattern length decreases. Since the main objective of the test points are to make the undetected faults in the circuit easier to detect, excessively large test lengths are not required.

Some of the benchmark circuits contain redundant faults which are not made testable by the control point insertion procedure. To observe the effectiveness of the test point insertion procedure on the testable faults, the experiment was performed using a new fault set created by removing the redundant faults from the collapsed fault set. Table 5.12 shows the results of this experiment. The coverage is given in terms of the new fault set.

More than half of the circuits attain 100% coverage of all detectable faults after test points are added. The CPU time is also reduced due to the smaller fault set.

Circuit	Test Points			Coverage		CPU
	obs	ctl	co	before	after	(s)
C432	2	0	0	0.9940	1.0000	82
C499	2	0	0	0.9956	1.0000	104
C880	0	0	0	1.0000	1.0000	2
C1355	2	1	0	0.9977	1.0000	220
C1908	4	1	0	0.9976	0.9996	220
C2670	49	5	1	0.9081	0.9951	803
C3540	56	8	0	0.9675	0.9933	1026
C5315	53	0	0	0.9946	0.9996	722
C6288	16	0	0	0.9949	0.9937	1330
C7552	65	11	1	0.9626	0.9973	2110
C349	0	0	0	0.9973	0.9973	16
C420	1	1	0	0.9799	1.0000	75
C444	4	2	0	0.9870	0.9965	74
C526	1	0	0	0.9995	1.0000	56
C526n	0	0	0	1.0000	1.0000	1
C641	0	0	1	0.9923	1.0000	14
C713	19	2	1	0.9600	0.9674	190
C820	0	0	0	1.0000	1.0000	2
C832	8	0	0	0.9931	1.0000	170
C838	5	6	1	0.8849	1.0000	188
C953	0	0	0	1.0000	1.0000	2
C1196	1	1	0	0.9987	1.0000	125
C1238	14	4	0	0.9744	0.9982	250
C1423	11	0	1	0.9929	1.0000	231
C1494	10	0	0	0.9958	1.0000	229
C5378	24	3	0	0.9895	0.9981	678
C9234	100	18	0	0.9073	0.9926	5281
C13207	72	9	0	0.9883	0.9980	2469
C15850	100	36	0	0.9537	0.9954	133406
C38417	100	40	0	0.9756	0.9987	23846
C38584	100	51	0	0.9536	0.9838	6741

Table 5.11: Coverage before and after test point insertion (102400 patterns)

Circuit	Test Points			Coverage		CPU
	obs	ctl	co	before	after	(s)
C432	0	0	0	1.0000	1.0000	1
C499	0	0	0	1.0000	1.0000	1
C880	0	0	0	1.0000	1.0000	2
C1355	0	0	0	1.0000	1.0000	4
C1908	1	0	0	0.9995	1.0000	23
C2670	3	0	1	0.8848	1.0000	41
C3540	1	1	0	0.9985	1.0000	49
C5315	0	0	0	1.0000	1.0000	10
C6288	0	0	0	1.0000	1.0000	27
C7552	11	6	1	0.9613	0.9969	143
C349	6	0	0	0.9943	0.9943	5
C420	1	1	0	0.9419	1.0000	9
C444	5	1	0	0.9705	0.9958	8
C526	1	0	0	0.9982	1.0000	7
C526n	0	0	0	1.0000	1.0000	1
C641	0	0	1	0.9829	1.0000	6
C713	19	1	1	0.9208	0.9707	18
C820	2	1	0	0.9941	1.0000	12
C832	10	1	0	0.9747	1.0000	18
C838	5	5	1	0.8611	0.9743	25
C953	3	2	0	0.9898	1.0000	14
C1196	2	1	0	0.9815	0.9992	18
C1238	21	3	0	0.9328	0.9948	29
C1423	14	2	0	0.9861	1.0000	25
C1494	10	0	0	0.9920	1.0000	25
C5378	5	1	0	0.9912	1.0000	57
C9234	100	21	0	0.8359	0.9860	689
C13207	12	14	0	0.9439	0.9999	1272
C15850	45	14	2	0.9446	0.9975	734
C38417	87	26	1	0.9296	0.9988	28108
C38584	68	19	2	0.9847	0.9997	845

Table 5.12: Test point insertion with redundant faults removed

5.5.1 Comparison with Other Methods

In general, probabilistic methods for test point insertion require much less CPU time than simulation based methods. An extra step of fault simulation, however, is required to determine the exact coverage. The Fault Injection Method used in this thesis as the last stage of the test point insertion process is based on fault simulation and provides exact fault detectability results which can be used to determine the final fault coverage.

The results in Table 5.12 can be compared with those obtained by a probabilistic method [STS91] shown in Table 5.13. The results in Table 5.13 show the fault coverage obtained using 32000 random patterns with redundant faults removed. The number of test points is similar, however circuit C7552 did not reach 100% coverage in Table 5.12.

The results for C2670 in Table 5.12 are better than those in Table 5.13 in terms of the number of observation points and control points used.

	Test Points		Coverage	
Circuit	obs	ctl	before	after
C2670	7	3	0.8821	1.0000
C7552	2	18	0.9611	1.0000

Table 5.13: Test point insertion results from [STS91]

5.5.2 Manual Placement of Test Points

The analysis tools provide a list of lines at which to place test points. These points may not all be necessary or there may be other choices which will have the same effect. The user may take these points as suggestions which indicated the areas in which controllability and observability need to be enhanced.

It is possible that the placement of test points can be improved by the user. In Table 5.12 it is shown that the coverage of testable faults in C2670 can be improved to

100% with only 3 observation points at lines 1556, 1555 and 894 and 1 control/observe point at line 1559. These test points are placed in one specific area of the circuit illustrated in Figure 5.8.

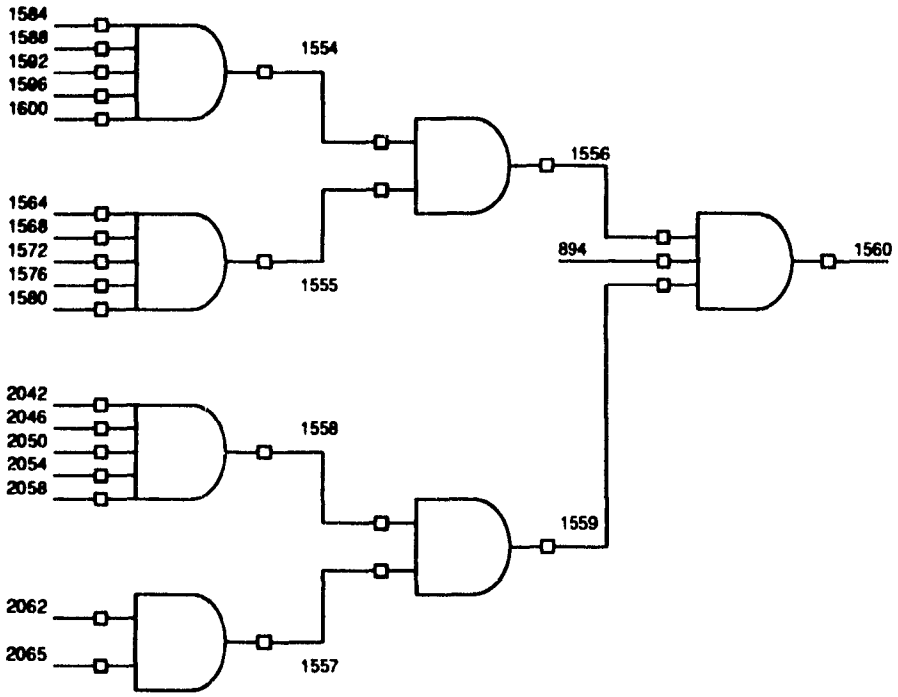


Figure 5.8: Part of circuit C2670

With a small amount of experimentation, it was determined that by adding OR type control points at lines 1554, 1555 and 1558 and observation points at lines 1556 and 1559 it was possible to obtain 100% fault coverage under all the sequences of 10240 patterns that were tried.

In this case, more control points are used instead of observation points. The placement of control and observation points are related. Adding more control points may reduce the number of observation points needed. This relationship would complicate any scheme that attempted to minimize the number of test points needed.

The method presented in this thesis does not attempt to minimize the number of test

points. It relies on the use of control points and observation points to independently improve the testability of the circuit.

Chapter 6

Conclusions and Future Work

As the marketplace demands more quality the need for higher coverage tests becomes apparent. With the growing popularity of built-in self-test (BIST) comes an increased reliance on random pattern testing. In order to keep random pattern test lengths low and fault coverage high, circuits must be made more testable. Test point insertion is a technique that can accomplish these goals and can be automated.

In this thesis a framework for test point insertion in combinational circuits was developed. Two methods based on fault simulation and probabilistic methods for determining both estimates and exact values of fault detectability were presented. Simulations were performed to verify the estimates.

An implementation of these methods was used to analyze a series of benchmark circuits. The placement of test points was automatically determined from the results of this analysis. Results of these experiments show an improvement in testability can be obtained.

The test point analysis was done using a hybrid of probabilistic and simulation based methods in an effort to achieve a compromise between analysis time and accuracy. The final stage of observation point analysis was based on fault simulation to provide exact

fault detectability results and thus allow the fault coverage to be determined without the need for additional fault simulations. This technique can be used to augment other probabilistic test point insertion methods by replacing the final stage of fault simulation with the observation point analysis. This will provide both an exact fault coverage value and may also improve the placement of observation points.

This research leads to several areas of potential future work. The combining of different methods for test point analysis requires more investigation in the hopes of obtaining a better trade off between simulation time and accuracy. There will always be room for the development of more heuristics that attempt to qualify the relationship between control points and observation points.

Bibliography

[AF81] V.K. Agarwal and A. Fung. Multiple fault testing of large circuit by single fault test sets. *IEEE Transactions on Computers*, C-30(11):855–865, November, 1981.

[AMM83] M. Abramovici, P.R. Menon, and D.T. Miller. Critical path tracing - an alternative to fault simulation. In *Proceedings of the 20th Design Automation Conference*, pages 214–220, 1983.

[And80] H. Ando. Testing vlsi with random access scan. *Proceedings COMPCON, Spring 1980*, pages 50–52, 1980.

[Arm72] Douglas B. Armstrong. A deductive method for simulating faults in logic circuits. *IEEE Transactions on Computers*, C-21(12):464–471, May 1972.

[AS86] K.J. Antreich and M.H. Schulz. Fast fault simulation in combinational circuits. In *Proceedings of the International Conference on Computer Aided Design*, pages 330 -333, November 1986.

[AS87] K.J. Antreich and M.H. Schulz. Accelerated fault simulation and fault grading in combinational circuits. *IEEE Transactions on Computer Aided Design*, CAD-6(5):704 -712, 1987.

[AS88] Vishwani D. Agrawal and Sharad C. Seth. *Test Generation for VLSI Chips.* Computer Society Press, Washington, D.C., 1988.

[BART82] C.C. Beh, K.H. Arya, C.E. Radke, and K.E. Torku. Do stuck fault models reflect manufacturing defects? In *Proc. International Test Conference*, pages 35 -42, Washington D.C., 1982.

[BBK89] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential benchmark circuits. In *Proc. IEEE Int. Symp. Circuits Systems*, pages 1929–1934, 1989.

[BF85] F. Brglez and H. Fujiwara. A neutral netlist of 10 combinational benchmark circuits and a target translator in fortran. In *Proc. IEEE Int. Symp. Circuits Systems*, pages 663-698, June 1985. Special Session on ATPG and Fault Simulation.

[BHP+71] W.G. Bouricius, E.P. Hsieh, G.R. Putzolu, J.P. Roth, P.R. Schneider, and C.J. Tan. Algorithms for detection of faults in logic circuits. *IEEE Transactions on Computers*, C-20(12):1258-1264, Nov. 1971.

[Bot86] P. S. Bottorff. Test generation and fault simulation. In T. W. Williams, editor, *VLSI Testing*, volume 5 of *Advances in CAD for VLSI*, chapter 2, pages 29-64. Noth-Holland, 1986.

[Brg84] F. Brglez. On testability analysis of combinational networks. *Proceeding International Symposium on Circuits and Systems*, pages 221-225, May, 1984.

[BT86] A.J. Briers and K.A.E. Totton. Random pattern testability by fast fault simulation. In *Proc. International Test Conference*, pages 274-281, 1986.

[Cox91] H. Cox. *On Necessary and Nonconflicting Assignments in Algorithmic Test Pattern Generation*. PhD dissertation, McGill University, Department of Electrical Engineering, January 1991.

[Eld59] R.D. Eldred. Test routines based on symbolic logical statements. *Journal of the ACM*, 6:33-36, 1959.

[EW77] E.B. Eichelberger and T.W. Williams. A logic design structure for lsi testing. *Proc. 14th Design Automation Conf.*, pages 462-468, June 1977.

[FKY89] S. Funatsu, M. Kawai, and A. Yamada. Scan design at nec. *IEEE Design and Test*, 6(3):50-57, June 1989.

[FS83] H. Fujiwara and T. Shimono. On the acceleration of test generation algorithms. *IEEE Trans. Comp.*, C-32:1137-1144, Dec. 1983

[FT82] H. Fujiwara and S. Toida. The complexity of fault detection problems in combinational logic circuits. *IEEE Transactions on Computers*, C-31(6):555-560, June, 1982.

[Fuj85] H. Fujiwara. *Logic testing and design for testability*, page 15. MIT Press, Cambridge, Mass., 1985.

[GJ78] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., New York, NY, 1978.

[Goe81] P. Goel. An implicit enumeration algorithm to generate tests for combinational logic circuits. *IEEE Transactions on Computers*, C-30(3):215–222, March, 1981

[Gol80] L.H. Goldstein. Scoap: Sandia controllability/observability program. In *Proc. 17th Design Automation Conference*, pages 190–196, 1980.

[Hay74] J.P. Hayes. On modifying logic networks to improve thier diagnosability. *IEEE Transactions on Computers*, C-23(1):56–62, January 1974.

[HF74] J.P. Hayes and A.D. Friedman. Test point placement to simplify fault detection. *IEEE Transactions on Computers*, c-23(7):727–735, July 1974.

[HHB89] HHB, Mahwah, New Jersey. *Intelligen Documentation Package*, 1989.

[HM86] J.L.A. Hughes and E.J. McCluskey. Multiple stuck-at coverage of single stuck-at test sets. In *Proc. International Test Conference*, pages 368–374, Washington, D.C., September 1986.

[Hon78] S.J. Hong. Fault simulation strategy for combinational logic networks. *Digest 8th International Symposium on Fault-Tolerant Computing Systems*, pages 96-99, June, 1978.

[IB89] Vijay S. Iyengar and Daniel Brand. Synthesis of pseudo-random pattern testable designs. In *Proc. International Test Conference*, pages 501–508, 1989.

[JA84] S.K. Jain and V.D. Agarwal. Stafan: An alternative to fault simulation. *Proc. 21st Design Automation Conf.*, pages 18–23, 1984.

[JB87] J. Jacob and N.N. Biswas. Gtbd faults and lower bounds on multiple fault coverage of single fault test sets. In *Proc. International Test Conference*, pages 849-855, Washington D.C., September 1987.

[Kau68] W.H. Kautz. Fault testing and diagnosis in combinational digital circuits. *IEEE Transactions on Computers*, C-17(4):352–366. April 1968.

[KMZ79] B. Koenemann, J. Mucha, and G. Zwiehoff. Built-in logic block observation techniques. *Proc. 1979 IEEE Test Conf.*, pages 37–41, 1979.

[Koe86] S. Koeppe. Modeling and simulation of delay faults in cmos logic circuits. In *Proc. International Test Conference*, pages 530–536, 1986.

[MC71] E.J. McCluskey and F.W. Clegg. Fault equivalence in combinational logic networks. *IEEE Transactions on Computers*, C-20(11):1286–1294, Nov. 1971.

[McC85] E.J. McCluskey. Built-in self-test techniques. *IEEE Design and Test*, 2(2):21 28, April 1985.

[MR90a] F. Maamari and J. Rajski. The dynamic reduction of fault simulation. In *Proc. International Test Conference*, pages 801 808, September 1990.

[MR90b] F. Maamari and J. Rajski. A method of fault simulation based on stem regions. *IEEE Transactions on Computer-Aided Design*, 9(2):212 220, February 1990

[PM75] K.P. Parker and E.J. McCluskey. Probabilistic treatment of general combinational networks. *IEEE Transactions on Computers*, C-24(6):668 670, June 1975.

[PRM90] A. Pancholy, J. Rajski, and L.J. McNaughton Empirical failure analysis and validation of fault models in cmos vlsi. In *Proceedings International Test Conference*, pages 938–947, Washington, D.C., 1990.

[RC90] J. Rajski and H. Cox. A method to calculate necessary assignments in algorithmic test pattern generation. In *Proceedings International Test Conference*, pages 25–34, Washington, D.C., September 1990

[SA88] M.H. Schulz and E. Auth. Advanced automatic test pattern generation and redundancy identification. *Proc. FTCS-18*, pages 30 35, June 1988.

[Ses65] S. Seshu. On an improved diagnosis program. *IEEE Trans. Elec. Comp.*, EC-14(2):76–79, 1965.

[SM72] D.R. Schetz and G. Metze. A new representation for faults in combinational digital circuits. *IEEE Transactions on Computers*, C-21(8):858 866, August 1972.

[SPA85] S.C. Seth, L. Pan, and V.D. Agrawal. Predict probabilistic estimation of digital circuit testability. *Fault-tolerant Comp Symp. (FTCS-15) Digest of Papers*, pages 220–225, June 1985.

[SR74] K.K. Saluja and S.M. Reddy. On minimally testable logic networks. *IEEE Transactions on Computers*, C-32(1):552 554, 1974.

[STS91] Bernhard H. Seiss, Pieter M. Trouborst, and Michael H. Schulz. Test point insertion for scan-based bist. In *European Test Conference*, 1991.

[STW89] George Swan, Yatin Trivedi, and David Wharton. Crosscheck - a practical solution for asic testability. In *Proc. International Test Conference*, pages 903–908, 1989.

[SYKK91] Y. Savaria, M. Youssef, B. Kaminska, and M. Koudil. Automatic test point insertion for pseudo-random testing. In *Proc. IEEE Int. Symp. Circuits Systems*, pages 1960-1963, 1991.

[Sys91] Systems Performance Evaluation Cooperative. *SPEC Newsletter*, September 1991.

[TS75] E.W. Thompson and S.A. Szygenda. Digital logic simulation in a time based, table-driven environment part 2. parallel fault simulation. *Computer*, 8(3):38–49, March 1975.

[TS88] K. Totton and S. Shaw. Self-test: The solution to the vlsi test problem? *IEE Proceedings*, 135, Pt. E(4):190 195, July 1988.

[UB74] E.G. Ulrich and T. Baker. Concurrent simulation of nearly identical digital networks. *Computer*, 7(4):39 -44, April 1974.

[Wad78] R.L. Wadsack. Fault modeling and logic simulations of cmos and mos integrated circuits. *Bell Syst. Tech. Jour.*, 57:1449–1474, May–June 1978.

[WEF+85] J.A. Waicukauski, E.B. Eichelberger, D.O. Forlenza, E. Lindbloom, and T. McCarthy. A statistical calculation of fault detection probabilities by fast fault simulation. *Proc. ITC-85*, pages 779-784, 1985.

[WLR87] J.A. Waicukauski, E. Lindbloom, B.K. Rosen, and V.S. Iyengar. Transition fault simulatoin. *IEEE Design and Test of Computers*, 4:32–38, April 1987.

[WP79] T.W. Williams and K.P. Parker. Testing logic networks and design for testability. *Computer*, pages 9–21, October 1979.

[WP83] T.W. Williams and K.P. Parker. Design for test 'ity—a survey. *Proc. IEEE*, 71:98 112, Jan. 1983.

[Wun85] H.J. Wunderlich. Protest: A tool for probabilistic testability analysis. In *Proc. 22nd Design Automation Conference*, pages 204–211, 1985.