Diagnostic Test Pattern Generation and Fault Simulation for Stuck-at and Transition Faults

by

Yu Zhang

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 4, 2012

Keywords: DATPG, Fault Simulation, Fault Diagnosis

Abstract

In VLSI testing we need *Automatic Test Pattern Generator* (ATPG) to get input test vectors for *Circuit Under Test* (CUT). Generated test sets are usually compacted to save test time which is not good for failure diagnosis. Physical defects in circuits are modeled by different *Fault Models* to facilitate test generation. Stuck-at and transition fault models are widely used because of their practicality. In this work a *Diagnostic Automatic Test Pattern Generation* (DATPG) system is constructed by adding new algorithmic capabilities to conventional ATPG and fault simulation programs. The DATPG aims to generate tests to distinguish stuck-at fault pairs, i.e., two faults must have different output responses. This will help diagnosis to pin point the failure by narrowing down the fault candidates which may be the reason for this particular failure. Given a fault pair, by modifying circuit netlist a new single fault is modeled. Then we use a conventional ATPG to target that fault. If a test is generated, it is guaranteed to distinguish the fault pair in the original circuit. A fast diagnostic fault simulation algorithm is implemented to find undistinguished fault pairs from a fault list for a given test vector set. To determine the quality of the test vector set generated by DATPG, we use a proposed *Diagnostic Coverage* (*DC*) metric, which is similar to *Fault Coverage* (FC). The diagnostic ATPG system starts by first generating conventional fault coverage vectors. Those vectors are then simulated to determine the DC, followed by repeated applications of diagnostic test generation and simulation. We observe improved *DC* in all ISCAS'85 benchmark circuits.

To distinguish between a pair of transition faults, we need to find a test vector pair (LOC or LOS type) that produces different output responses for the two faults. By adding a few logic gates and one modeling flip-flop to the circuit under test (CUT), we extend the ability of the previous DATPG system to target transition faults. Given a transition fault pair, this ATPG model either finds a distinguishing test or proves the faults to be equivalent. The number of fault pairs that needs to be targeted by the ATPG is greatly reduced after diagnostic fault simulation. Experimental results show improved *DC* and shorter CPU time compared to a previous work [38].

Although not demonstrated in this thesis, the proposed DATPG system can be used for multiple or mixed fault models as long as there is an ATPG and fault simulator for targeted faults. This work thus allows any conventional fault detection ATPG tools to be enhanced for diagnosis purposes without significantly increasing their complexity.

Acknowledgments

This work is completed with help from a lot of people. First and foremost, I owe my deep gratitude to my advisor Dr. Vishwani Agrawal. When I ran out of ideas, he always sparkled me with suggestions. When I made mistakes he patiently guided me back on track. He is always encouraging and supportive, which makes the PhD work not intimidating at all. I always feel that I cannot thank him enough.

I thank Dr. Adit Singh and Dr. Victor Nelson for being my committee member and giving me a lot of inspiring suggestions during my presentations of research. I learned a lot from Dr. Bogdan Wilamowski's enlightening and delightful class talks, such as Advanced Numeric Methods, etc. which serve as part of the fundamental knowledge for my work. All his students, including me, are thankful for his humorous teachings. I very much appreciate that Dr. Sanjeev Baskiyar served as the university reader in spite of his packed schedule.

Also I am very grateful to my colleagues, without whom my study and research at Auburn would not have been enjoyable. Wei, Kim, Ashfaq, Chao, Nitin, Manish, Priya, Jia, Lixing, Suraj and many more were there to help me in various ways. Some help may seem small or trivial but even ocean is formed from creeks, isn't it?

Last but not least, my undying gratitude goes to my caring parents and loving wife.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

In the realm of modern VLSI designs, testing of the product constitutes a major part of the total cost. For example, a 300mm wafer can produce about 700 1cm^2 chips. An estimation of the manufacturing cost can be broken down like this [41, 66]:

- Material cost: ∼5%
- Fab amortization cost (over $3 billion/fab) and fab operational cost: ∼25%
- Personnel cost: ∼20%
- Packaging cost: ∼10%
- Testing cost: ∼40%

Diagnosis plays a crucial role in cutting down testing cost and improving yield. To test a circuit, physical defects are modeled with different *Fault Models*. A common objective of testing is to detect all or most modeled faults. Although fault coverage (detected faults divided by total number of faults) has a somewhat nonlinear relationship with the tested product quality or defect level (failing parts per million), for practical reasons fault coverage continues to be a measure of the test quality [20].

Most test generation systems are built around core ATPG (Automatic Test Pattern Generation) algorithms [20] for (1) finding a test vector for a target fault and (2) simulating faults to find how many have been detected by a given vector. The system then attempts to find tests of high fault coverage because the primary objective is fault detection, i.e., presence or absence of faults.

In some test scenarios we must diagnose or identify the fault, making the test intent different from the original detection coverage. In practice, detection tests with high coverage may not have adequate diagnostic capability. One often uses tests for multiple fault models [91, 98, 99] or multiple tests for the same model [15]. Basically, we generate tests that are redundant for fault detection and then hope that they will provide better diagnostic capability. To reduce the excess tests we may resort to optimization or removal of unnecessary tests [39, 82, 83].

Advanced VLSI technology employs finer features that are comparable or even smaller than the wavelength used in lithography. This leads to larger process variability, leading to non-classical fault models that the tests should be tuned to. As a result, the initial yield ramp-up or characterization phase where defect identification is required has assumed greater importance. The term non-classical faults refers to faults that do not conform to the simple stuck-at fault model widely used in theory and industry practice. Characterization phase refers to the pre-production testing where faults produced by the fabrication process are diagnosed, analyzed and, if possible, eliminated. Details on these terms and processes can be found in textbooks on electronic testing [4, 20, 43].

In this work we are dealing with classical fault and non-classical fault diagnosis. Classical fault refers to the single stuck-at fault model [4, 20, 43], which assumes a single faulty signal line permanently set to 0 or 1, i.e. the logic value on that line cannot be toggled. We express this fault as line s-a-0 or s-a-1. A variety of other fault models are termed as non-classical faults. Some examples are bridging fault (coupling between two or more signals), stuck-open or stuck short fault (a transistor permanently open or short), transition fault (delayed transition of a signal), path delay fault (extra delay on a clocked signal path), and I_{DDQ} fault (excess leakage or quiescent current) [20]. Details about different fault models can be found in Chapter 2.

1.1 Problem Statement and Contributions

Goals of this work are:

- Generate additional test vectors to distinguish fault pairs for multiple fault models.
- Develop efficient diagnostic fault simulation algorithm to get undistinguished fault pairs and DC.
- Define diagnostic coverage (DC).
- Implement above algorithms into a DATPG system.
- Minimize test set with DATPG and compare to ILP based method.

A contribution of this work is in providing basic core algorithms for a diagnostic test generation system with the capability to handle multiple fault models. Given a pair of faults, we should either find a distinguishing test or prove that the faults are equivalent in a diagnostic sense; diagnostic equivalence of two faults implies that faulty functions at all outputs of a multi-output circuit are identical [76]. Two faults in the pair may belong to different fault models. The new exclusive test algorithm of Chapter 4 and 5 represents a non-trivial improvement in computational efficiency over previously published work [6, 38]. We provide a diagnostic coverage metric in Chapter 4 similar to the detection fault coverage, such that a 100% diagnostic coverage means that each modeled fault is distinguished from all other faults. This definition leads to efficient diagnostic fault simulation of Chapter 4, which is another core algorithm. A fault simulator is an essential tool for obtaining meaningful tests. Years of research have produced highly efficient fault simulation algorithms and programs [20]. Both exclusive test generation and diagnostic fault simulation algorithms are specifically devised to take full advantage of the highly developed existing tools for stuck-at faults. The key features of the diagnostic algorithm presented here are (1) it accepts fault detection data from any conventional fault simulator, thus benefiting from the efficiency of a mature program, (2) fault dropping is used to delete diagnosed faults from the list of faults as fault simulation progresses, and (3) for a given set of input vectors it provides fault coverage (FC), diagnostic coverage (DC), and necessary data for a fault dictionary. In Chapter 4 a diagnostic test generation system is presented which combines the core algorithms and the new coverage metric into an ATPG system.

We further extend the ability of the system to target transition faults and bridging faults, by modeling them with stuck-at faults. Actually, as long as the faulty behavior can be mapped to certain logic (combinational or sequential), our ATPG system can generate distinguishing test vectors for that modeled fault.

With these diagnostic test vectors we can improve the test minimization algorithm proposed in [82, 83, 98, 99]. Using a dedicated diagnostic test set instead of an N-detect test set further reduces the CPU time.

1.2 Organization of this Dissertation

In Chapter 2, we present the basics of fault testing and diagnosis. Terminologies are defined and explained. Previous work are summarized. Overview of VLSI diagnosis process are presented in Chapter 3. In Chapter 4, we discuss in detail the proposed DATPG system. Chapter 5 will expand the capability of the DATPG system to deal with transition faults. Finally in Chapter 6 we give the conclusion and discuss the possibility to target arbitrary fault models.

Chapter 2

Background and Overview of DATPG

While VLSI testing is considered a mature science, that maturity applies to fault detection. Detection tests, used in high-volume production, identify the device as good or faulty without identifying the specific fault present. Larger process variation now places greater emphasis on fault location in the pre-production or characterization test phase. Most work on diagnostic tests has appeared only since mid to late 1990s. Ad-hoc definitions of diagnostic coverage or diagnostic resolution exist but there is no accepted metric for diagnostic tests, similar to fault coverage for detection tests. Algorithms for an exclusive test to distinguish between two faults, functional equivalence identification, or fault dictionary optimization are known to be NP-hard. They require practical heuristic solutions before they can be incorporated into a diagnostic test generation system. The proposed research will advance the state of the art in diagnostic testing through algorithms and tools for non-classical faults that are relevant to today's nano-scale technologies.

This chapter deals with the basics of VLSI testing. First I will give a brief introduction about different type of VLSI circuits, faults models, ATPG algorithms, and DFT techniques, etc. Fundamentals described here will help the reader understand the proposed algorithms.

2.1 Combinational and Sequential Circuits

We can generally divide digital circuits into 2 subcategories: combinational and sequential. Combinational circuit means there's no memory element in the circuit. The logic is purely combinational. Figure 2.1 shows a common combinational circuit with several primary inputs (PI) and primary outputs (PO). If input vectors (logic 0s and 1s) are applied

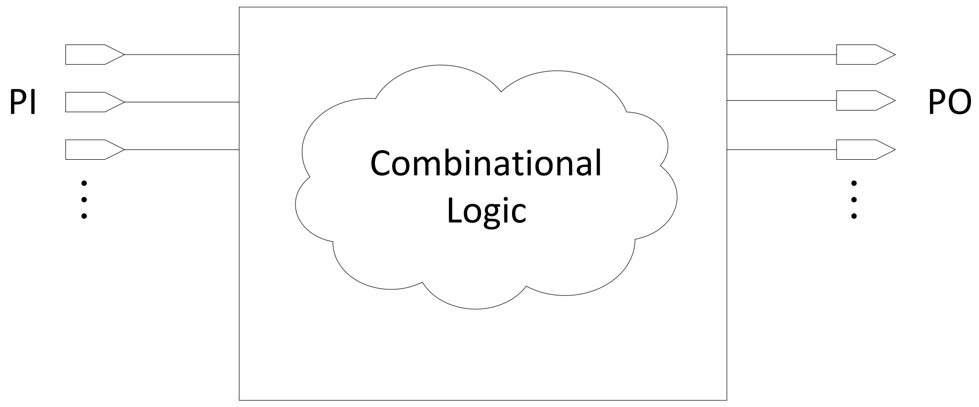


Figure 2.1: Illustration of a combinational circuit.

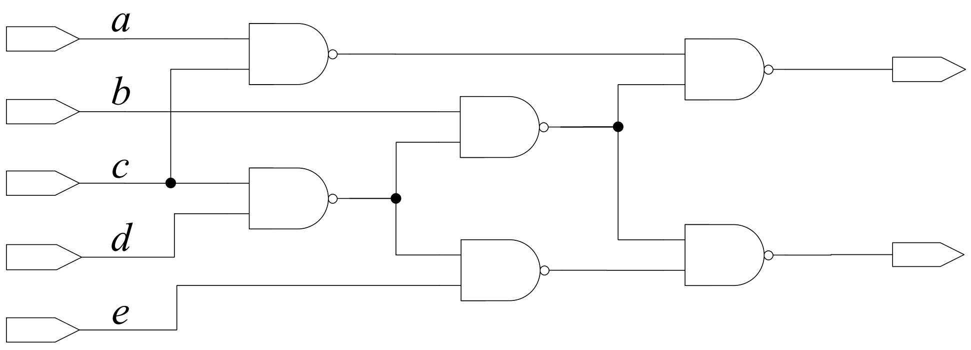


Figure 2.2: ISCAS'85 combinational benchmark circuit c17.

at PIs, output responses can be observed immediately at POs (after any gate delays). Figure 2.2 shows a simple example of a combinational circuit. If we apply a test vector "01010" at the primary inputs we should get "11" if the circuit functions correctly.

Actually nearly all digital designs are sequential, i.e. there are flip-flops and/or other state holding elements inside the circuit. Introducing memory elements such as D flip-flops and latches [65] makes it possible to reuse logic and hold state, which can extend the functionality of the circuit tremendously. Figure 2.3 shows a diagram for typical sequential logic.

2.2 VLSI Testing

Testing is the process to verify that the fabricated VLSI circuit functions as expected or designed. Figure 2.4 shows the principle of typical digital testing. Circuit/Device under

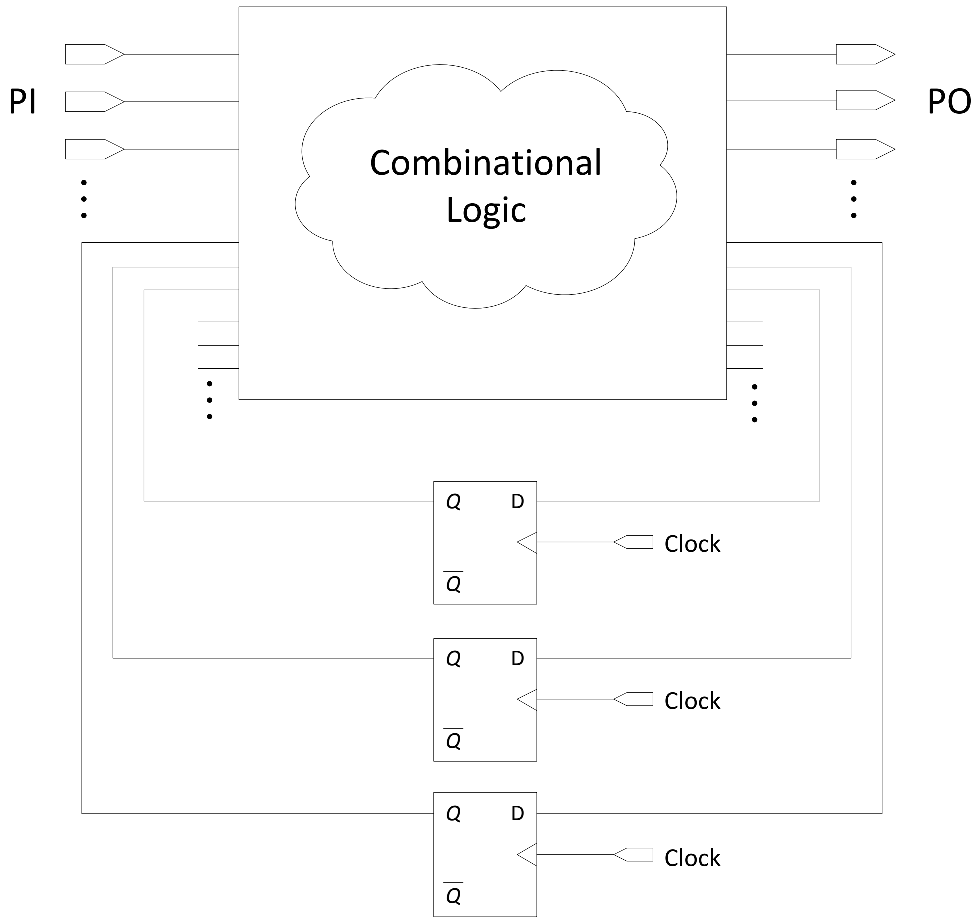


Figure 2.3: Illustration of a sequential circuit.

test (CUT/DUT) is exercised by applying test signals to its inputs. Output responses are observed and compared. If there's no mismatch between observed response and stored correct response, the CUT is considered good, otherwise it will be labeled as bad. The input data to the CUT is referred to as the test pattern or test vector. Usually tests applied during testing are for fault detection. The purpose of a detection test is to identify the CUT as faulty of fault-free. But in diagnosis, additional test patterns maybe applied, aiming to find the cause/location of the failure.

Currently the available commercial silicon fabrication technology reduces the feature size down to 22nm [1]. Process variation has greater and greater impact on product quality. A manufactured circuit may develop various kinds of defects during and after the fabrication process [57].

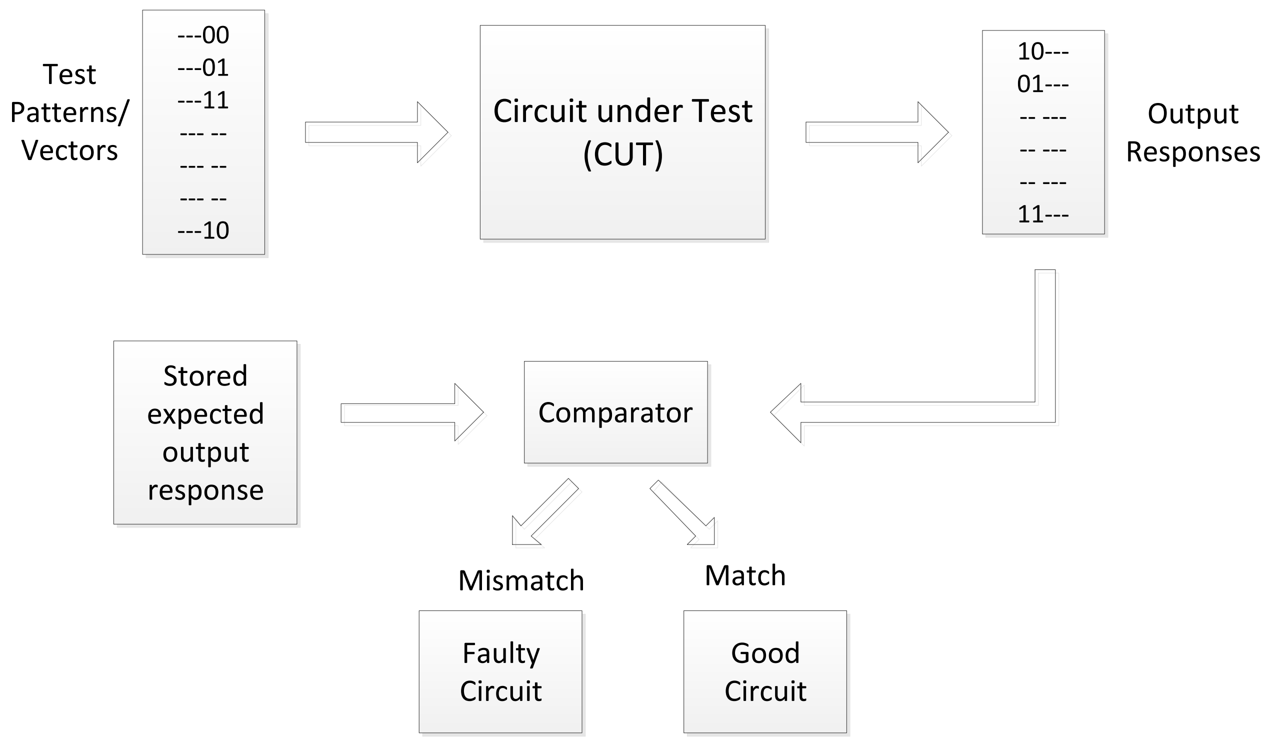


Figure 2.4: Basic test process.

There are several different types of testing [20]:

- Verification testing, characterization testing, or design debug
  - Verifies correctness of design and correctness of test procedure – may require correction of either or both
- Manufacturing testing
  - Factory testing of all manufactured chips for parametric and logic faults, and analog specifications
  - Burn-in or stress testing
- Acceptance testing (incoming inspection)
  - User (customer) tests purchased parts to ensure quality

In this work we focus on manufacturing test. For details about other testing scenarios, interested readers can refer to several text books [4, 20, 26, 43, 64, 65, 87, 97]. Input test vectors are usually generated by an Automatic Test Pattern Generator (ATPG). Another popular area in testing/test pattern generation is Built-in Self-test (BIST) [87]. For chips with BIST structure all or part of the input test patterns are generated by on chip logic. The idea is to let chips test themselves to cut down the testing cost since external Automatic Test Equipment (ATE) is very expensive. The cost of a tester for current VLSI chips can easily go over \$20M [20, 87].

In manufacture testing, at different fabrication stages there are different kinds of tests. Before the chip is packaged, wafer sort or probe test is conducted. Gate threshold voltage is measured to make sure it is within acceptable range. Other monitored parameters include polysilicon sheet resistance, poly field threshold voltage, gate delay, currents, etc. These measurements are called parametric testing. Another key testing scenario is the functional/structure test. A full functional test is usually impractical. For example if we want to exercise all possible inputs for a simple 64 bit ripple carry adder, 2^{129} test patterns are needed (two 64-bit data plus one bit carry). A 1GHz automatic test equipment (ATE) will take 10^{22} years to test it [20]. On the other hand, structure testing is widely used in practice whose purpose is to verify the topology of the circuit under test (CUT). Based on the assumption that the correctness of the design is verified before manufacturing (though design verification is another huge problem to solve in the VLSI world), if we can test that every node in the fabricated chip is toggling as simulated/expected then we can be sure about the quality of the final product.

The goal of testing is to correctly separate good chips from bad ones. In reality the test process itself is not perfect. It is possible that bad chips are not picked out and good chips are labeled as bad. The former are called "test escapes" and the latter are called "yield loss". Both may bring serious economic loss [20]. To ensure high product quality, diagnosis plays a key role in improving testing procedures and may point out the direction for better design methodology in future products.

2.3 Fault Modeling

To conduct structural testing first we need to model real world defects with mathematical fault models. A good fault model should accurately reflect the behavior of a physical defect, and it should be computationally efficient for simulation [87].

2.3.1 Stuck-at and Redundant Fault

Stuck-at fault [31, 33] is the most widely used fault model. As the name implies, stuck-at fault is a signal line (gate input/output) stuck-at logic "0" or "1", i.e. it cannot change its value during operation. In most literature stuck-at fault indicates gate level stuck-at fault; so does this writing. Figure 2.5 shows an example of stuck-at fault [87]. The output of AND3 is stuck at 0, meaning that no matter what the inputs are, the output of AND3 is always 0 after power up. Correct logic value is shown in black before "/" and the faulty value is shown in red after "/". The logic function is a simple multiplexer and it can be expressed with Boolean algebra as:

$$Z = A\bar{S} + BS \tag{2.1}$$

The corresponding Karnaugh map [65] is shown in Figure 2.6. The reason to have redundant logic introduced into the multiplexer is to eliminate hazard. Figure 2.7 shows an example of hazard condition [2]. It illustrates a logic race condition, showing how the delays incurred in each element affect the final output. Interested reader can verify that after introducing additional AND3 in Figure 2.5 race condition is eliminated.

Redundancy and re-convergent fanout shown in Figure 2.5 brings undetectable faults (called redundant fault). For example, the s-a-0 fault at the output of AND3 can never be

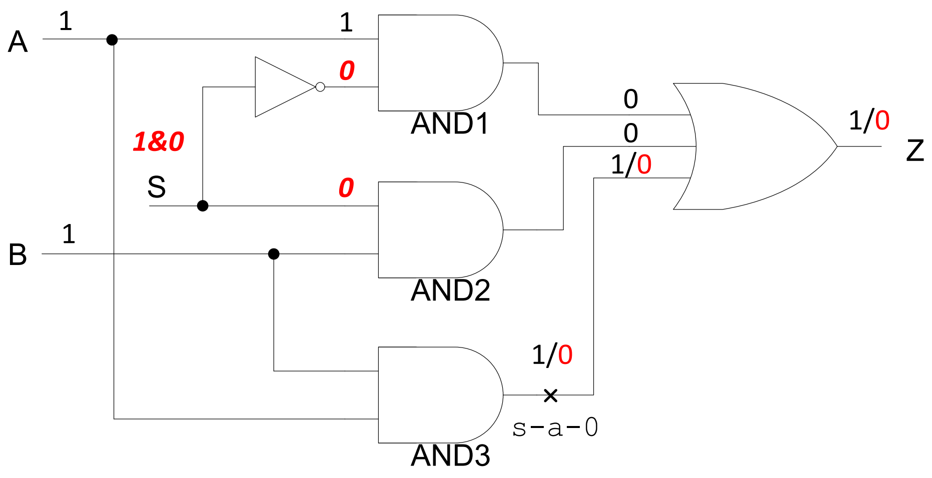


Figure 2.5: A multiplexer with s-a-0 fault and redundant logic to eliminate hazard [87].

	AB			
S	00	01	11	10
0	0	0	1	1
1	0	1	1	0

Figure 2.6: Karnaugh map for example of Figure 2.5. Items enclosed in red oval indicates the redundant logic (AND3 in Figure 2.5).

detected. To detect the fault, both inputs of AND3 need to be 1 and the first 2 inputs of OR gate must be 0 (in order to observe at primary output), and this will cause conflict conditions for signal S. S has to be 1 ***and*** 0, which is impossible. Thus s-a-0 on AND3 is a redundant fault. Redundancy identification is very important in conventional ATPG and our DATPG system. It will influence fault coverage and diagnostic coverage, which will be discussed in detail in the following chapters. Redundant fault does not affect the function of the circuit since its existence has no impact on output values (assume there is no hazard, i.e. no race condition).

Even though many physical defects do not manifest as stuck at fault, due to its effectiveness to detect real failures, stuck-at fault continues to serve as a "default" fault model in VLSI testing [47].

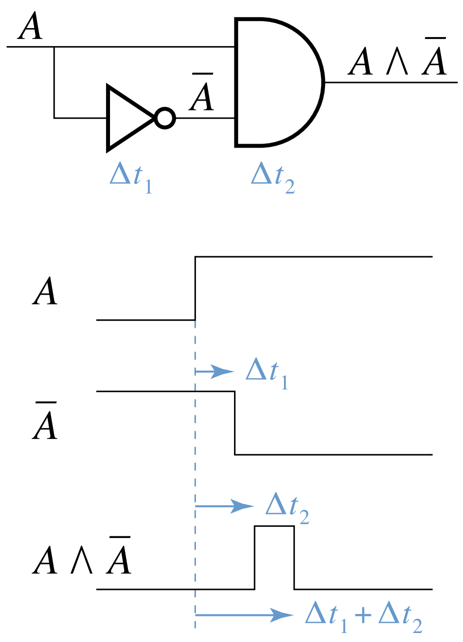


Figure 2.7: Race condition in a logic circuit. Here, Δt_1 and Δt_2 represent the propagation delays of the logic elements. When the input value (A) changes, the circuit outputs a short spike of duration $(\Delta t_1 + \Delta t_2) - \Delta t_2 = \Delta t_1$ [2].

2.3.2 Transition Fault Model

Timing related fault models are also widely used. Because of the growing complexity of modern VLSI designs it is more and more difficult to meet timing. Clock tree synthesis and static timing analysis are part of the techniques dealing with timing issues [64]. Two types of delay faults are commonly used. One is path delay fault, modeling accumulated excessive delay along a certain path. The other is transition fault, which assumes that excessive delay lumps at a gate input or output [20]. Since transition fault has similar behavior, as well as similar complexity, compared to stuck-at fault in testing, it also enjoys a widespread popularity.

2.3.3 Other Fault Models

The transistor level stuck-at model may be more accurate than gate level stuck-at fault model. It is also more complex. A single (or multiple) transistor can be stuck at *on* or *off* state, causing excessive current flow (stuck-short/on) or memory effect (stuck-open/off) [87, 20]. Since this model does not make any noticeable improvement in testing quality it has not enjoyed widespread success compared to stuck-at fault, which has much lower complexity [51].

Another fault model which is gaining popularity is called bridging fault. It models the defect in the interconnect of modern chips. There are several types of bridging faults: wired-AND, wired-OR and Dominant, etc. Wired-AND means two signal lines form an AND gate, i.e. if one line get a value of “0”, the other line will be forced to “0”. Similarly wired-OR causes “0” of one line to be overridden by the other line if it has a value of “1”. Dominant bridging fault indicates one line dominates the other no matter what value the other line has. More accurate bridging models, such as dominant-AND/dominant-OR have also been proposed [87].

IDDQ faults are those causing excessive current during steady state (e.g. transistor stuck-on is one possible cause of an IDDQ fault).

There are several other fault models that draw attention in VLSI testing. For example open fault model, which targets interconnect opens/breaks, disconnected vias, can result in state-holding, intermittent, and pattern-dependent fault behaviors [51].

2.4 Design for Test (DFT)

It is quite common for today’s digital VLSI designs to have millions of gates. Even with the single stuck-at fault model, trying to test every input/output of every gate is a challenging task. As discussed before, to exhaustively test each functionality of the design is impossible. A commonly used testing strategy is to structurally test every signal line (input/output of logic gate), to see whether they toggle correctly as simulated. To do this we need two prerequisites: controllability and observability. The former indicates that a test pattern/sequence should be able to set the targeted line to a desired value. And the latter means we should be able to observe the behavior of a line from a primary output or an observation point such as a scan flip-flop, which will be discussed below.

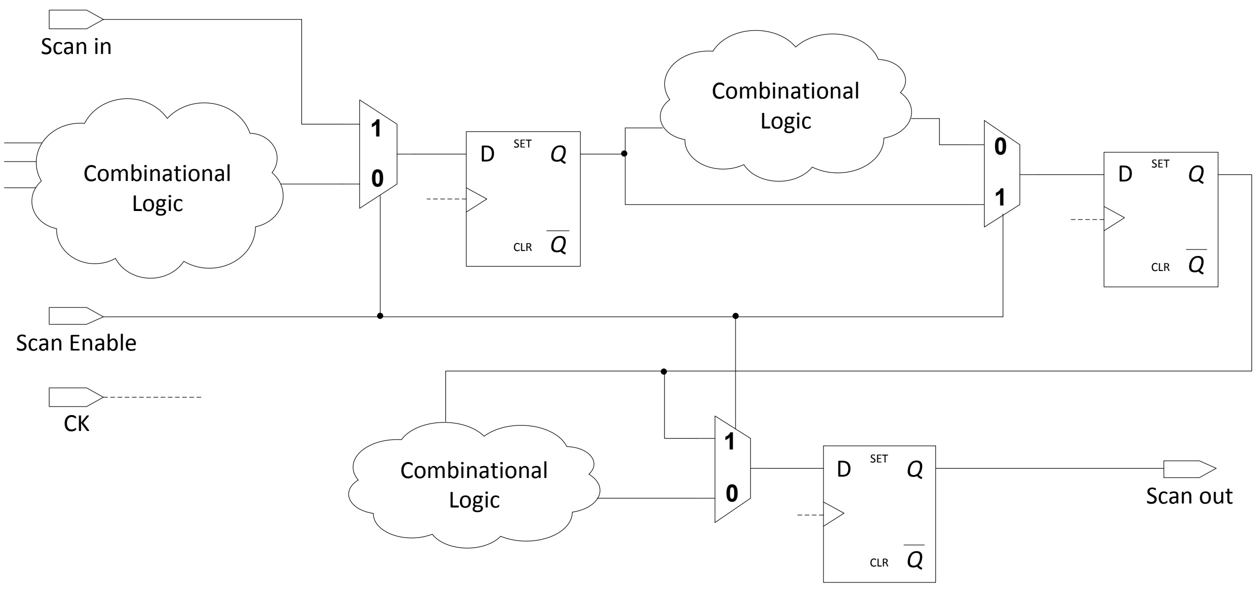


Figure 2.8: Scan design.

Currently, nearly all digital designs integrate scan flip-flops. Figure 2.8 shows the idea of scan design. Basically, all or most flip-flops are stitched together using different methods. One way to implement the scan flip-flop is to use multiplexers as shown in Figure 2.8. With scan design we can shift in any test pattern from a scan-in pin and observe the output values captured in flip-flops from a scan-out pin. The area overhead is not significant. And only several additional pins are needed (scan-in, scan-out, scan-enable, etc.), which are very valuable resources in digital design. State-of-the-art VLSI chips have billions of transistors and thousands of input/output pins. It is very difficult to add dedicated pins just for testing.

Quite often, even with scan chains implemented, some part of the circuit is still uncontrollable/unobservable. Ad-hoc techniques are applied in these situations to increase the fault coverage to nearly 100%. Nowadays people put much emphasis on reliability of electronic devices, thus near 100% fault coverage, along with low DPPM (defective parts per million), is usually required. To achieve this, control and/or observation points are inserted into the circuit which do not influence the functionality but make it easier to test.

Another popular DFT technique is Built-in Self-test (BIST). Test patterns are generated internally inside the CUT and output responses are compacted into a pass-fail bit or signatures. This means a chip can test itself without outside stimulus, thus greatly reducing the cost of testing, and ATEs are no longer needed.

In reality any single DFT technique is not sufficient to achieve near 100% coverage while maintaining a reasonable cost. Most designs implement a combination of several DFT techniques. For example BIST is used for initial screening and then ATE is used to apply additional deterministic test patterns for further testing.

2.5 ATPG and Fault Simulation

Automatic Test Pattern Generation is another key area in VLSI testing. Its primary role is to generate test patterns with as much coverage as possible. Lots of algorithms have been proposed since the early years of VLSI design, such as D-algorithm, PODEM, FAN, TOPS, etc. Introduction of these algorithms can be found in many text books [4, 20, 26, 43, 87].

An ATPG usually comes with a fault simulator whose goal is to simulate generated test patterns to find undetected faults, calculate coverage, etc. Any ATPG system will benefit a lot from a fast/efficient fault simulator.

Chapter 3

VLSI Diagnosis

Many diagnosis techniques have been proposed through the years [51]. In this work we use the dictionary based method as demonstration, though our diagnostic test generation will also benefit other diagnosis methods.

3.1 Introduction of Diagnosis Procedure

The purpose of diagnosis is to identify the cause of a failing chip, i.e., the nature and location of the defect. To hunt down the culprit causing the failure, a straightforward way is to go through every "household", using physical inspection equipment, such as scanning electron microscopes, particle beams, infrared sensors, liquid crystal films, laser scanner, etc [22, 51]. If the process is successful we may get a photograph of the "crime scene". It may be missing or extra metal conductor, resistive via, short/open transistor, cross talk between neighboring wire, and so on. In [14, 67] there are several "crime scene" pictures for open/short circuit failures obtained by SEM (Scanning Electron Microscope) or TEM (Transmission Electron Microscope).

Because of the large scale of VLSI designs it is impossible to search through the entire design with physical inspections. It is necessary to perform fault diagnosis to narrow down the possible defect locations. Diagnosis algorithms can be roughly divided into two categories: effect-cause fault diagnosis and cause-effect fault diagnosis [3, 51, 80].

Effect-cause diagnosis begins with the syndrome of the failing chip. A syndrome is the observed faulty behavior of the chip under test. Various path-tracing methods are applied to narrow down the logic locations (e.g. cross section of 2 logic cones). Several algorithms have been proposed in previous works such as structural pruning [96], back-trace [97], inject and evaluate [69], etc. In cause-effect diagnosis it is usually assumed that there can be only one fault which is causing the problem. This kind of assumption is not necessary in effect-cause diagnosis. Although it has some advantage in handling a CUT with multi-defects, the diagnosis results tend to be pessimistic and imprecise [80].

In cause-effect diagnosis, simulated faulty behaviors are stored/computed before hand. Once a faulty response of a CUT is captured during testing, this observed response is compared with the stored values. If there is a close or exact match, the CUT is diagnosed at the first stage. Equipped with the location/fault type information, the next stage is design tracing back to find weaknesses (timing/weak driving signal/large load/etc.) or physical inspection to dig deeper. For cause-effect diagnosis usually there are some assumptions to limit the mathematical complexity within a manageable extent. One popular assumption is single fault, i.e. there is only one fault causing the failure. Since a fault model itself is just an approximation of a real physical defect, quite often the observed faulty behavior of a CUT does not match the simulated behavior based on the assumed fault model. Additional algorithms are developed to deal with this problem [49]. Once a fault model is chosen (multiple fault models can be used, but one at a time), a basic diagnosis process can be performed by comparing the signature of the failing CUT with the simulated signature for each modeled fault. In [51] signature is defined as “the data representation of the response of a defective circuit to a diagnostic test set”. For example a signature of a stuck-at 0 fault on a primary input is the list of failing vectors that detect this fault and the outputs at which the incorrect/unexpected value is captured.

3.2 Fault Dictionary

Many previously proposed diagnosis methods rely on a fault dictionary [11, 70, 73, 84, 90, 94]. In this section we demonstrate the construction of different types of dictionaries.

Figure 3.1 shows a benchmark circuit with a stuck-at 1 fault (f1) on the top interconnect. Several test vectors (t1, t2, t3) are applied to detect the fault. All the faulty responses are

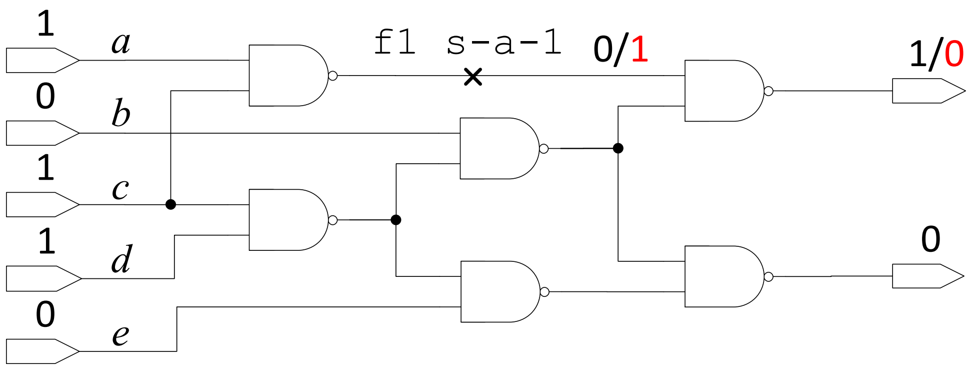


Figure 3.1: c17 with a stuck-at-1 fault.

Table 3.1: Observed output responses for fault s-a-1 in c17.

Faults	Output responses		
	t1 10110	t2 11110	t3 10101
fault free	10	10	11
f1	00	00	01
f2	10	00	11
f3	10	10	10
f4	10	10	01

recorded in table 3.1 with four faults injected, one at a time. Comparing the fault free response to the faulty responses we can get signatures for faults. In table 3.2 the "fault free" row has a signature of all 0s, indicating that observed bits are matching simulated/expected outputs. A signature bit of "1" means there is a mismatch on that output. In this example f1 (s-a-1) has a signature of "10" for test vector 1 (t1), meaning that a mismatch on the first output is captured when applying t1. If we look closer we can see that faulty behavior of f1 is only observed on the first output for all test vectors. In diagnosis this information can be used to narrow down the possible locations of the fault(s) (fan-in cone of the first output pin).

Table 3.2 shows one way to construct a full response dictionary. In a simple diagnosis scenario once a faulty response is observed we can look in the dictionary for a matching entry. If an entry is found then the corresponding fault is known. The type and location of the fault can be used to guide physical inspection to pin-point the real defect on silicon,

Table 3.2: Full response dictionary.

Faults	Signature t1	t2	t3
fault free	00	00	00
f1	10	10	10
f2	00	10	00
f3	00	00	01
f4	00	00	10

Table 3.3: Indexed full response dictionary.

Faults	Signature t1	t2	t3
f1	1	1	1
f2	-	1	-
f3	-	-	2
f4	-	-	1

otherwise it is impractical to use physical inspection through the entire chip. For large industry designs, a full response dictionary can become prohibitively large ($F * V * O$ bits), where F is the number of faults, V is the number of test vectors, and O is the number of outputs [80]. Dictionary compaction is another well researched area [25, 68, 50, 39]. For example we can use an index instead of binary bits for a dictionary. Table 3.3 shows an index dictionary converted from Table 3.2. Since all possible output patterns are equal to or smaller than $2^n - 1$, where n is the number of outputs, the largest index needed is $min(2^n - 1, highest\ number\ of\ faults\ detected\ by\ any\ test\ vector)$. In reality faults in a same logic cone tend to produce identical output responses for a given vector set, so that the largest index is usually much smaller than highest number of faults detected by any test vector [80]. In Table 3.3 detection at the first output is assigned index “1”, detection at the second output is assigned index “2”. The indexed dictionary can be stored as {f1: [(t1, 1), (t2, 1), (t3, 1)], f2: [(t2, 1)], f3: [(t3, 2)], f4: [(t3, 1)]}, and this is the one we use for our DATPG system in next chapter.

To further reduce the size of a dictionary, another approach is to use a pass-fail dictionary. As the name suggests, pass-fail dictionary only stores pass or fail status of a fault for

Table 3.4: Pass-fail dictionary.

Faults	Signature t1	t2	t3	Index
f1	1	1	1	1
f2	0	1	0	2
f3	0	0	1	3
f4	0	0	1	3

all applied vectors. Table 3.4 shows an example of a pass-fail dictionary. Fault f1 fails all 3 test vectors, thus having a signature of "111" ("1" indicating a fail). The corresponding indexes are shown in the right side of the table. Failing all three tests are assigned an index of 1. For every fault we can calculate the index like this: start from 1; increase by 1 as long as the signature of current fault is unique (i.e. previous faults do not have same signature). If the signature is a repetition of previous ones, the same index is assigned.

Obviously this compaction is achieved at the sacrifice of resolution. For example fault f3 and f4 have the same signature in the pass-fail dictionary, thus they cannot be distinguished. In other words, if a CUT is passing t1 and t2 but failing t3, we cannot know which fault (f3 or f4) is causing the failure. But with a full response dictionary they are distinguished because they are detected at different outputs.

Chapter 4

Combinational Diagnostic Test Generation

In this chapter we begin to construct our DATPG system using the stuck-at fault model and indexed fault dictionary defined in previous chapters.

To better diagnose a failing CUT, the more unique output responses we get the more accurate a conclusion we can reach. The purpose of a DATPG system is to generate additional test vectors that target pairs of faults, which can produce different output responses for that pair, thus increasing the resolution of the diagnosis. The final fault candidate list can be narrowed down with such tests. Fault candidates are those faults that have same or similar signatures as observed signatures for a failing CUT.

Although nearly all circuits are sequential, in testing/diagnosis they are transformed into combinational logic through scan design (refer to figure 2.8) or other DFT techniques. Details about sequential logic diagnosis are presented in chapter 5.

4.1 Background

First we present some background about diagnostic test generation, including definitions of exclusive test and diagnostic coverage (DC).

Given a pair of faults, *Exclusive test* has been defined as a test that detects one fault but not the other [6] in this pair. Its primary intent is to distinguish between the fault pair. For a multiple output circuit, this definition is applied separately to each output. An exclusive test can detect both faults as long as they are not being detected at the same outputs. Perhaps a more appropriate term would be *distinguishing test*. However, following existing usage of the term, we will continue with *exclusive test*.

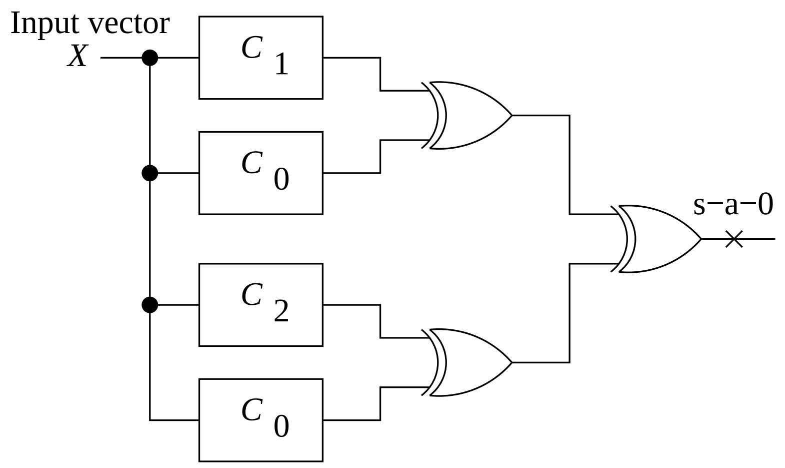


Figure 4.1: The general exclusive test problem.

This background is reproduced from a previous publication [6]. Consider a pair of faults, f_1 and f_2. An exclusive test must detect only one of these two faults. Figure 4.1 illustrates generation of an exclusive test. A fault-free digital circuit is shown as C_0. The other blocks C_1 and C_2 are the same circuit with faults f_1 and f_2, respectively. The circuit is assumed to be combinational and can have any number of inputs. For clarity, we will only consider single output functions for now. We show an example of a multiple output circuit at the end of this section. Any input vector that produces a 1 output in Figure 4.1, i.e., detects a s-a-0 fault at the primary output is an exclusive test for the fault pair (f_1, f_2). This is the Boolean satisfiability version of the exclusive test problem [103].

$$(C_0 \oplus C_1) \oplus (C_0 \oplus C_2) = 1 \qquad (4.1)$$

that can be simplified as:

$$C_1 \oplus C_2 = 1 \qquad (4.2)$$

This simplification, shown in Figure 4.2, implies that the two faulty circuit outputs differ for the exclusive test. There are two ways to generate exclusive tests. We can either modify a single-fault ATPG program [35] or modify the circuit and use an unmodified ATPG

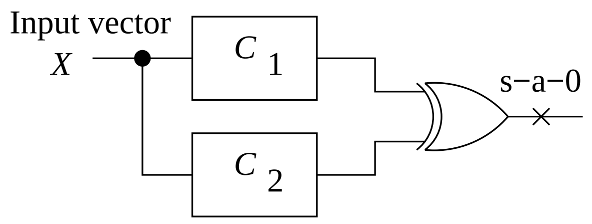


Figure 4.2: Simplified exclusive test.

program [6, 93]. In the previous work [6], the exclusive test generation problem was solved in a way similar to that showed in Figure 4.2. The fault pair was modeled as a single fault [48, 17] thus reducing the problem to that of single-fault ATPG in a twice as big a circuit. Figure 4.4 shows an example for generating an exclusive test for e1 (line e stuck-at 1) and b0 (line b stuck-at 0) in Figure 4.3. The model shown in Figure 4.4 is based on equation 4.3:

$$(C_0 \oplus C_0) \oplus (C_1 \oplus C_2) = 1 \tag{4.3}$$

Firstly e1 and b0 are modeled with the s-a-1 fault in Figure 4.4 [48] by adding three additional gates and doubling the original CUT. This means a test that detects s-a-1 in Figure 4.4 can also detect multiple faults e1 and b0 when they are both present in Figure 4.4 (without the added additional gates). Using any conventional single fault ATPG tool we can get a test for the s-a-1 fault. For example we get “0110” and this test fulfills the condition set in equation 4.3. The first part, $(C_0 \oplus C_0)$, represents the fault free response “0”. The second part $(C_1 \oplus C_2)$ indicates that when e1 exists in one copy of CUT and b0 exists in another copy of CUT, the two faulty CUT will produce different output responses for the generated test (because of the added XOR gate in Figure 4.4). We successfully generate an exclusive test for this fault pair at the expense of doubling the original CUT and adding several modeling gates to transfer multiple faults into a single fault. A serious limitation for this approach is that it can only generate tests for single output circuits. To extend the method for multiple output circuits a XOR (Exclusive-Or) tree is used to compact output

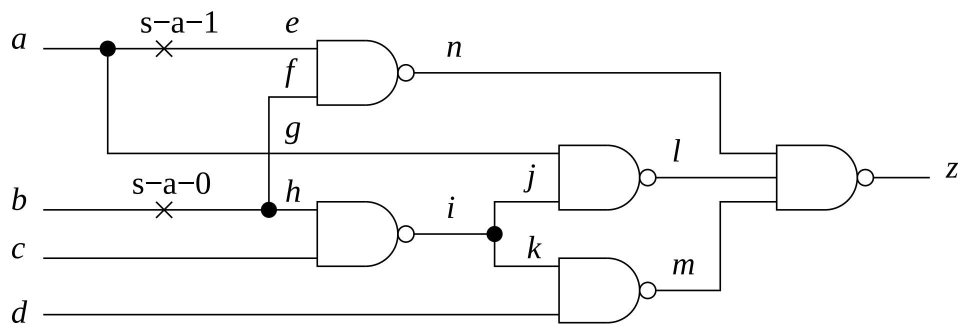


Figure 4.3: A circuit with exclusive test required for e s-a-1 and b s-a-0 [6].

responses to a single bit. An exclusive test for the modified single output circuit is also an exclusive test for the original multiple output circuit. Unfortunately the converse is not true [6].

Another way to enhance the 2 copy model to accommodate multiple output circuits is shown in Figure 4.5 [80]. Circuit C_1 is a CUT with fault f_1 inserted and C_2 is a CUT with fault f_2 inserted. By using an array of XOR gates and one multiple input OR gate we transformed the exclusive test generation problem into a detection test generation problem. If a test for s-a-0 at the output of the OR gate can be found, this test is the exclusive test for fault pair f_1 and f_2. This construction ensures that at least one output response differs between C_1 and C_2 in order to set the output line of OR gate to "1" to detect the s-a-0 fault.

Now we solved the exclusive test generation problem with only single stuck-at fault ATPG. One of the advantages of this approach is that various highly developed ATPG tools [88, 63, 24, 52] can be used. A previous paper proposed a specially designed ATPG tool [35] which targets differences between faults instead of single faults. Though the reported experimental results are good, its use is limited to fault pair distinguishing, and it cannot utilize the newly developed ATPG algorithms.

4.2 Boolean Analysis of New Exclusive Test Algorithm

We propose a new exclusive test generation algorithm which simplifies the DATPG problem to a single stuck-at ATPG problem. There is no need to double the circuit under

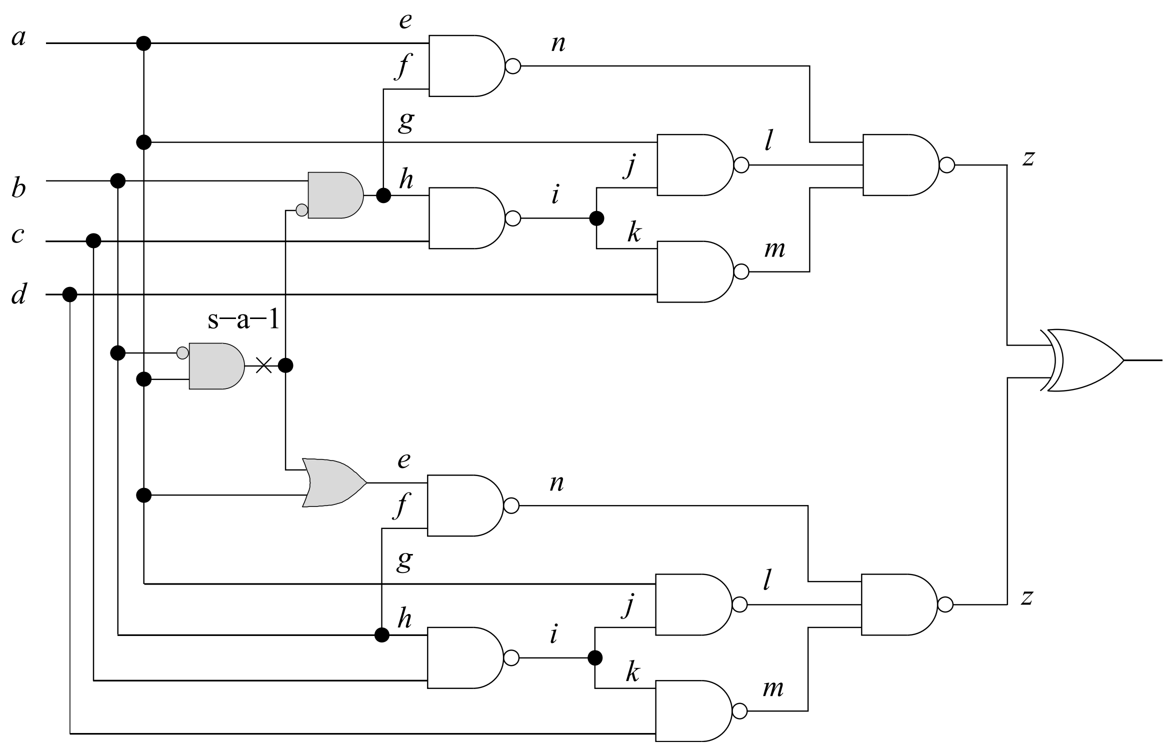


Figure 4.4: A two-copy ATPG model for generating exclusive test required for e s-a-1 and b s-a-0 [6].

test. The only modification to the CUT is inserting at most three logic gates (AND, OR, NOT) and an additional primary input pin. Then we use any conventional ATPG tool to target the s-a-0 or s-a-1 fault on the added input pin. If a test is found, this test is an exclusive test for the fault pair under investigation.

To get a deep understanding of the transformation of the previous DATPG model to our simplified model, Boolean algebra is used for deduction. In order to further simplify the solution shown in Figure 4.2 and equation 4.2, we first transform it into an equivalent single-fault ATPG problem shown in Figure 4.6. Here we have introduced a new primary input variable y. The function G in Figure 4.6 is clearly implemented as Shannon's expansion [20] about y with cofactors C_1 and C_2:

$$G(X, y) = \bar{y}C_1 + yC_2 \tag{4.4}$$

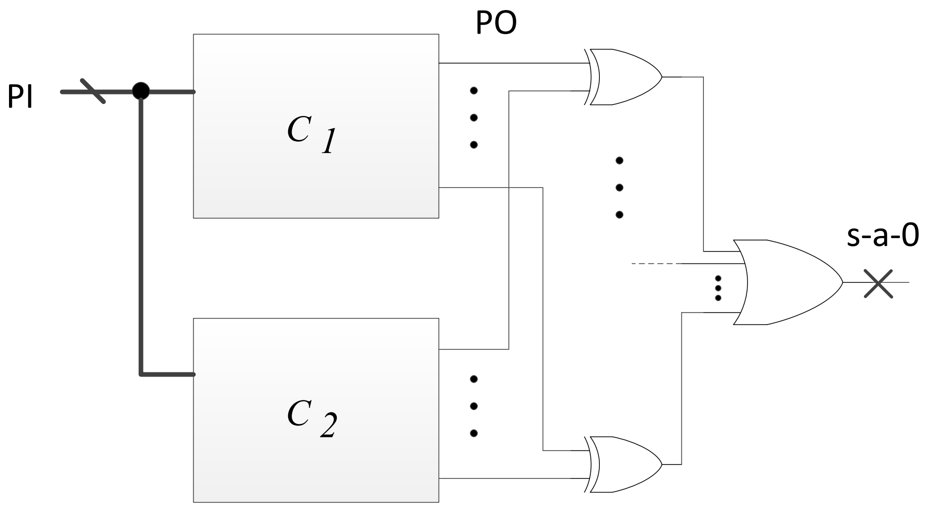


Figure 4.5: Two-copy ATPG model for multiple output circuit.

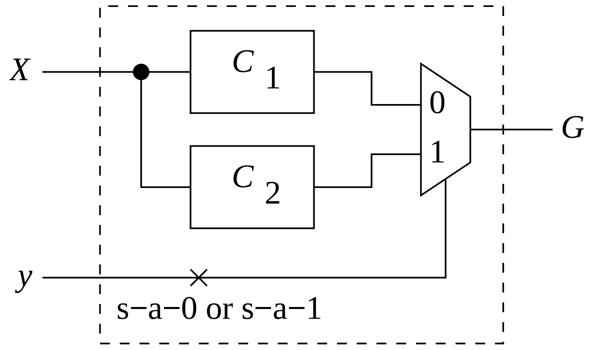


Figure 4.6: An ATPG circuit for exclusive test.

The condition for detecting s-a-0 or s-a-1 fault on y, using Boolean difference [20], is

$$\frac{\partial G}{\partial y} = C_1 \oplus C_2 = 1 \tag{4.5}$$

This is identical to the exclusive test condition of equation 4.2. Thus, we establish that a vector X that detects either s-a-0 or s-a-1 fault on y in the circuit $G(X, y)$ of Figure 4.6 will also detect the output s-a-0 fault in the circuit of Figure 4.2. We call G as *ATPG circuit*. It maps the given complex ATPG problem to a single fault ATPG problem. Next, we synthesize a compact circuit for $G(X, y)$.

Suppose fault f_1 is line x_1 s-a-a and fault f_2 is line x_2 s-a-b, where x_1 and x_2 are two signal lines in the circuit C. Fault variables a and b can assume any value 0 or 1. The primary input of C is a vector X that may or may not contain the two fault lines. We express the fault-free function using Shannon's expansion [20] as,

$$\begin{aligned} C(X, x_1, x_2) &= \bar{x}_1\bar{x}_2C(X,0,0) + \bar{x}_1x_2C(X,0,1) \\ &\quad x_1\bar{x}_2C(X,1,0) + x_1x_2C(X,1,1) \end{aligned} \tag{4.6}$$

Therefore, the cofactors of $G(X, y)$ are given by,

$$\begin{aligned} C_1 = C(X, a, x_2) &= \bar{a}\bar{x}_2C(X,0,0) + \bar{a}x_2C(X,0,1) \\ &\quad a\bar{x}_2C(X,1,0) + ax_2C(X,1,1) \\ C_2 = C(X, x_1, b) &= \bar{x}_1\bar{b}C(X,0,0) + \bar{x}_1bC(X,0,1) \\ &\quad x_1\bar{b}C(X,1,0) + x_1bC(X,1,1) \end{aligned} \tag{4.7}$$

Let us define the following variables:

$$x'_1 = \bar{y}a + yx_1 \quad \text{and} \quad x'_2 = \bar{y}x_2 + yb \tag{4.8}$$

We can clearly see that the function in equation 4.8 is multiplexing. Using the rules of Boolean algebra, such as absorption and consensus theorems [65], we obtain:

$$\bar{x}'_1\bar{x}'_2 = \bar{y}\bar{a}\bar{x}_2 + y\bar{x}_1\bar{b} \tag{4.9}$$

$$\bar{x}'_1x'_2 = \bar{y}\bar{a}x_2 + y\bar{x}_1b \tag{4.10}$$

$$x'_1\bar{x}'_2 = \bar{y}a\bar{x}_2 + yx_1\bar{b} \tag{4.11}$$

$$x'_1x'_2 = \bar{y}ax_2 + yx_1b \tag{4.12}$$

First we substitute C_1 and C_2 from equations 4.7 into equation 4.4 and then make use of equations 4.9 through 4.12, to obtain:

$$\begin{aligned} G(X,y) &= \bar{x}'_1\bar{x}'_2C(X,0,0) + \bar{x}'_1x'_2C(X,0,1) \\ &\quad x'_1\bar{x}'_2C(X,1,0) + x'_1x'_2C(X,1,1) \\ &= C(X,x'_1,x'_2) \end{aligned} \tag{4.13}$$

where the last result follows from the Shannon's expansion of the original circuit function C, given by equation 4.6, in which new variables x'_1 and x'_2 defined in equations 4.8 replace x_1 and x_2, respectively.

The synthesized circuit for $G(X,y)$, shown in Figure 4.7, is obtained by inserting two multiplexers, $a-mux$ and $b-mux$ controlled by a new primary input y, in the original circuit $C(X)$. Veneris et al. [93] arrive at a similar construction, though our derivation provides greater insight into the procedure. In practice, for any 0 or 1 value of variables a and b, each multiplexer simplifies either to a single gate or a gate with an inverter.

Let's try the new algorithm on the previous example circuit of Figure 4.3. We seek an exclusive test for two faults shown (e1, b0). The ATPG circuit for this problem is given in Figure 4.8 based on the above deductions. The logic shown with shading is obtained by simplifying the multiplexers of Figure 4.7 upon setting $a = 1$ and $b = 0$. The exclusive test found by a single fault ATPG is $a = 0,\ b = 1,\ c = 1,\ d = 0$. The signal values shown on lines are from five-valued D-algebra [20]. We arrive at same result compared to the method in the last section without doubling the CUT.

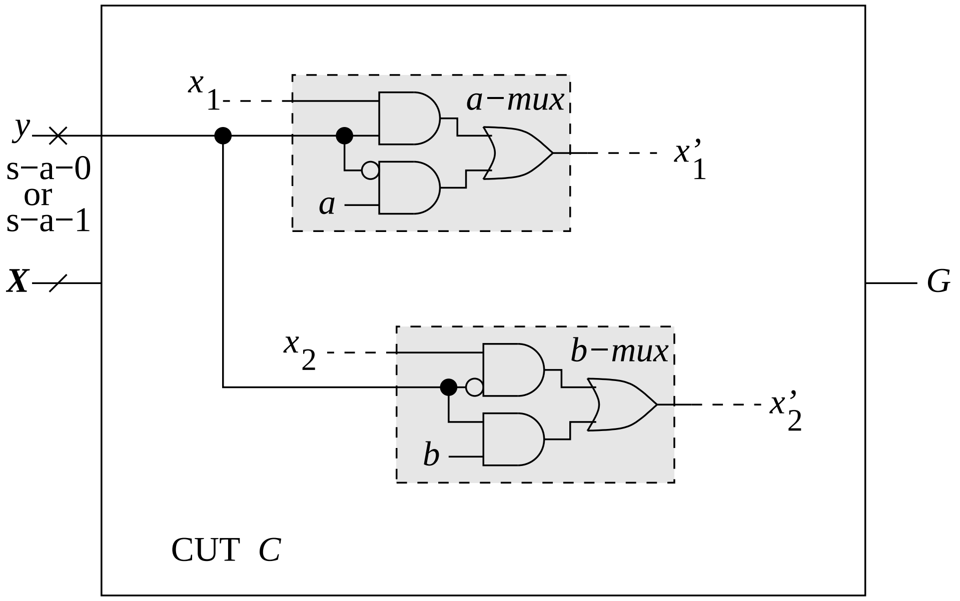


Figure 4.7: ATPG circuit with multiplexers inserted in CUT such that a test for s-a-0 or s-a-1 fault on y is an exclusive test for faults x_1 s-a-a and x_2 s-a-b.

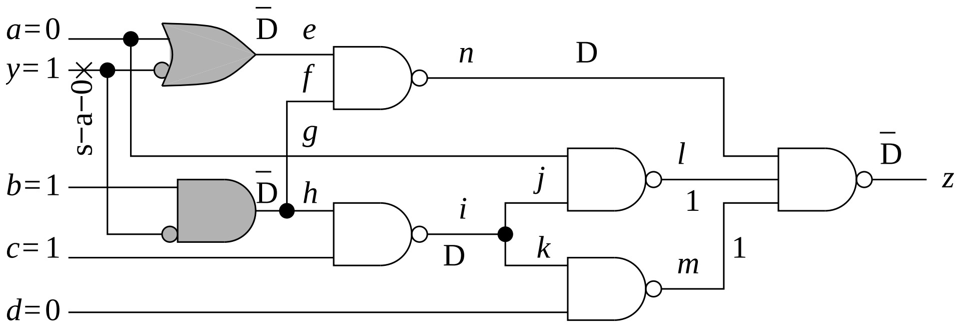


Figure 4.8: ATPG circuit for the exclusive test problem of Figure 4.3.

4.3 Generation of Exclusive Test

An exclusive test for two faults in a single output circuit must detect only one of those faults. In a multiple output circuit, that same condition must be satisfied at least on one output. Suppose, C_1 and C_2 blocks in Figure 4.2 each has n outputs. Previously we have to use n XOR gates such that the ith XOR receives the ith outputs from C_1 and C_2. All XORs feed into an n-input OR gate whose output contains the single s-a-0 faults to be detected (Figure 4.5).

In general, an exclusive test for a multiple output circuit can detect both targeted faults as long as they are not being detected on exactly the same outputs. Based on this extended definition the ATPG circuit derived in the previous section (Figure 4.6 and Figure 4.7) remains valid for multiple outputs. Figure 4.9 extends the ability of previous single output algorithm to accommodate two-output CUT. Similarly we can construct n-output ATPG circuits for exclusive test generation.

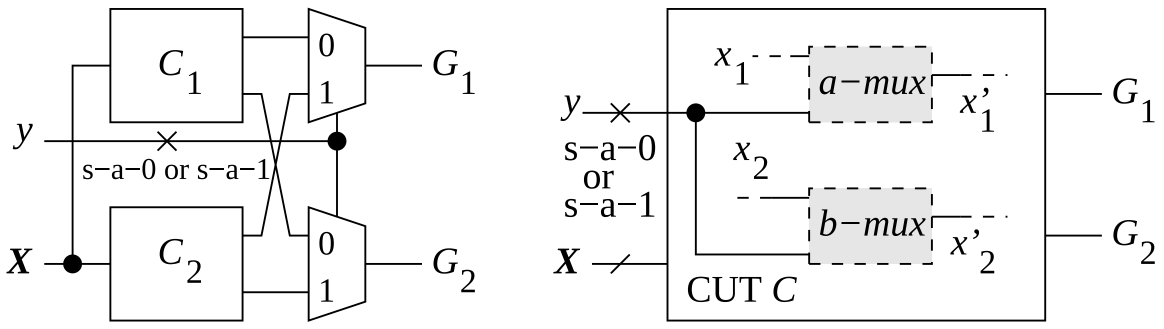
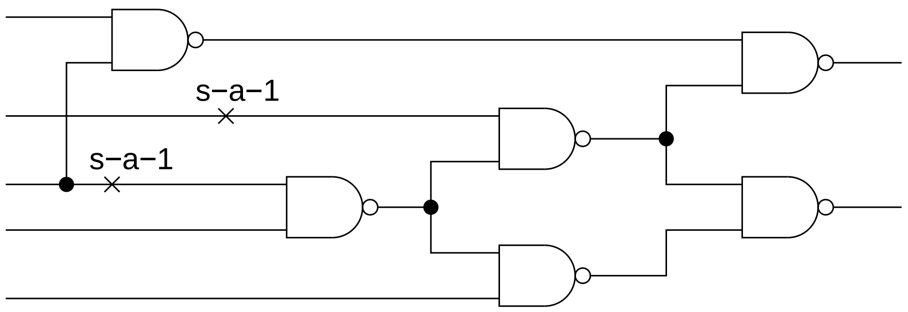


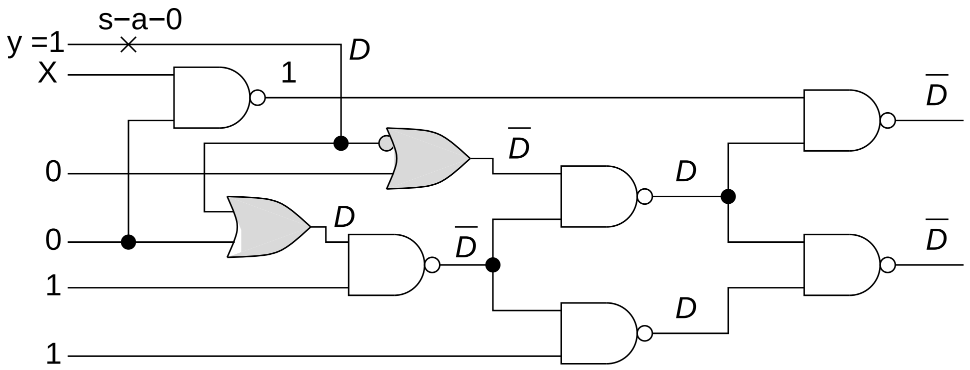
Figure 4.9: Exclusive test ATPG circuit for a two-output CUT.

Now let's look at a simple example to get a deeper understanding. Consider the problem of finding an exclusive test for two s-a-1 faults in the ISCAS'85 benchmark circuit c17 shown in Figure 4.10(a). The fault on signal y in the ATPG circuit of Figure 4.10(b) provides a test $X0011$. We can verify that this test detects both faults but at different outputs, hence it will distinguish between them.

Compared to [6], which need 2 copies of the CUT to distinguish between a pair of faults, we only use one copy. We insert 2 or 3 primary gates (AND, OR, NOT) and a new primary input (PI) to the netlist. Then any conventional ATPG program targets a stuck-at fault on the new PI. Either a test is found or the PI fault is proven to be redundant. In the former case, the test is an exclusive test that distinguishes between the fault pair. In the latter case, the faults in the pair are equivalent. Fault equivalence can be defined as: given two or more faults from a fault model, if their output responses are identical for all possible tests they are equivalent faults. So, in our modeled circuit, if the fault on the inserted PI is proven to be redundant, i.e. no test exists to detect this fault, the corresponding fault pair is equivalent.



(a) C17: Exclusive test problem for two s−a−1 faults.



(b) ATPG circuit and an exclusive test.

Figure 4.10: Exclusive test example for multi-output circuit C17.

Since the detection of this fault and distinguishing of the targeted fault pair are equivalent problems (proved in the previous section using Boolean algebra) fault equivalence checks can be transferred to redundancy identification. The advantage is that redundancy identification is a well researched area for conventional ATPG algorithms. We can use ATPG to do equivalence check to aid the diagnosis process without any additional effort. These diagnosis problems can be easily solved using highly developed testing tools. And the complexity of diagnosis is reduced to a level similar to that of fault testing. In [12, 13, 100] the author presents equivalence check techniques based on implication of faulty values and evaluation of faulty functions in cones of dominator gates of fault pairs. The major limitations are a special tool is needed and not all equivalence can be identified.

Table 4.1: Modified dictionary from Chapter 3.

Faults	Signature with (t1, t2)		Signature with t3
	t1	t2	t3
	10110	11110	10101
f1	1/10	1/10	1/10
f2	0/00	1/10	0/00
f3	0/00	0/00	1/01
f4	0/00	0/00	1/10

4.4 Diagnostic Coverage Metric

To generate diagnostic tests we need a coverage criterion to measure how good the generated test vectors are. This would be similar to the fault coverage (FC) used in conventional ATPG systems where fault detection is the objective. 100% FC means that all modeled faults are detected by the test set. In [6] *diagnostic resolution* (DR) is introduced to measure the quality of a given test set for fault diagnosis. DR is defined as:

$$DR = \frac{Total\ number\ of\ faults}{Number\ of\ syndromes} \tag{4.14}$$

Fault syndrome is the same as fault signature. DR represents the average number of faults per fault class (faults with the same syndrome/signature). A perfect DR of 1.0 is achieved if all faults have unique syndromes and all equivalent faults are identified (only one fault from an equivalent class is kept in the calculation, others are dropped). Here is a simple example for calculating DR. In Table 4.1 signatures before "/" are from a pass-fail dictionary and those after "/" are from a full-response dictionary. For the pass-fail dictionary we have 3 unique signatures: (111, 010, 001). Thus $DR = 4/3 = 1.33$. For full-response there are 4 different signatures: (101010, 001000, 000001, 000010), and $DR = 4/4 = 1.0$

In this work we propose a diagnostic coverage metric similar to DR but with a form similar to Fault Coverage (FC), making our DATPG system almost the same as a conventional ATPG system.

Since the core algorithm of previous sections generates exclusive/distinguishing tests for fault pairs, our first inclination was to specify coverage as the fraction of distinguishable fault pairs with respect to all possible fault pairs. This has been called diagnostic resolution (different than the DR we defined above) in several papers [21, 23, 95]. In [38] number of undistinguished fault pairs is used to indicate the diagnosability of a certain test set. Consider an example. For N faults, there will be $N(N-1)/2$ fault pairs. For a moderate size circuit, suppose $N = 10,000$, then there are 49,995,000 fault pairs. If our tests do not distinguish 5,000 fault pairs, then the coverage would be $(49,995,0005,000)/49,995,000 = 0.999$, which turns out to be highly optimistic. There is the additional problem of high complexity; for $N = 10^6$, which is common for industrial designs [58], the number of fault pairs is approximately half a billion. We, therefore, propose an alternative metric. For a set of vectors we group faults such that all faults within a group are not distinguishable from each other by those vectors, while each fault in a group is pair-wise distinguishable from all faults in every other group. In other words, faults in the same group have same signature/syndrome, while faults from different groups have different signatures. This grouping is similar to equivalence collapsing except here grouping is conditional to the vectors. If we generate a new vector that detects a subset of faults in a group then that group is partitioned into two subgroups, one containing the detected subset and the other containing the rest (note that if faults are detected at different outputs they are placed into different subgroups). Suppose, we have sufficient vectors to distinguish between every fault pair, then there will be as many groups as faults and every group will have just one fault (similar to perfect DR of 1.0). Prior to test generation all faults are in a single group we will call g_0. As tests are generated, detected faults leave g_0 and start forming new groups, g_1, g_2, . . . g_n, where n is the number of distinguishable fault groups. For perfect detection tests g_0 will be a null set and for perfect diagnostic tests, $n = N$, where N is the total number of faults. We define diagnostic coverage, DC, as:

$$DC = \frac{Number\ of\ detected\ fault\ groups}{Total\ number\ of\ faults} = \frac{n}{N} \tag{4.15}$$

Initially, without any tests, $DC = 0$, and when all faults are detected and pair-wise distinguished, $DC = 1$. Also, the numerator in equation 4.15 is the number of fault dictionary syndromes and the reciprocal of DC is actually diagnostic resolution (DR) defined previously. And DR can be viewed as the average size of a fault group that cannot be further diagnosed. Other metrics, such as diagnostic expectation [74, 101] that gives unnormalized and often large values and diagnostic power [21] that gives normalized but pessimistic values, need to be compared with DC defined here. For completeness of this discussion, detection fault coverage (FC) is:

$$FC = \frac{Number\ of\ detected\ faults}{Total\ number\ of\ faults} = \frac{N - |g_0|}{N} \tag{4.16}$$

To illustrate the point we made before about an alternative choice for a diagnostic metric, consider Figure 4.11. This example is based on the fault simulation results for ISCAS'85 benchmark circuit c432. The graph shows two metrics for 69 detection and diagnostic vectors for the c432 circuit reported in subsection 4.7.2 (Table 4.4). Vectors are generated using our DATPG system targeting 520 structurally collapsed single stuck-at faults (excluding 4 identified redundant faults). The dashed line shows the fraction of distinguishable fault pairs [21, 23, 95], which rises to almost 100% with about 40 vectors, although there are 900 undistinguished fault pairs as shown by the dropping curve (dot-dash) in the middle of the graph. The solid line is the DCC coverage computed from Table 4.4. When this coverage approaches 100% the undistinguished fault pair drops to 0. Thus, DC is a better metric than the fault-pair coverage. For different CUTs, since circuit size varies, using only the number of undistinguished fault pairs [38] is not good for comparison purpose.

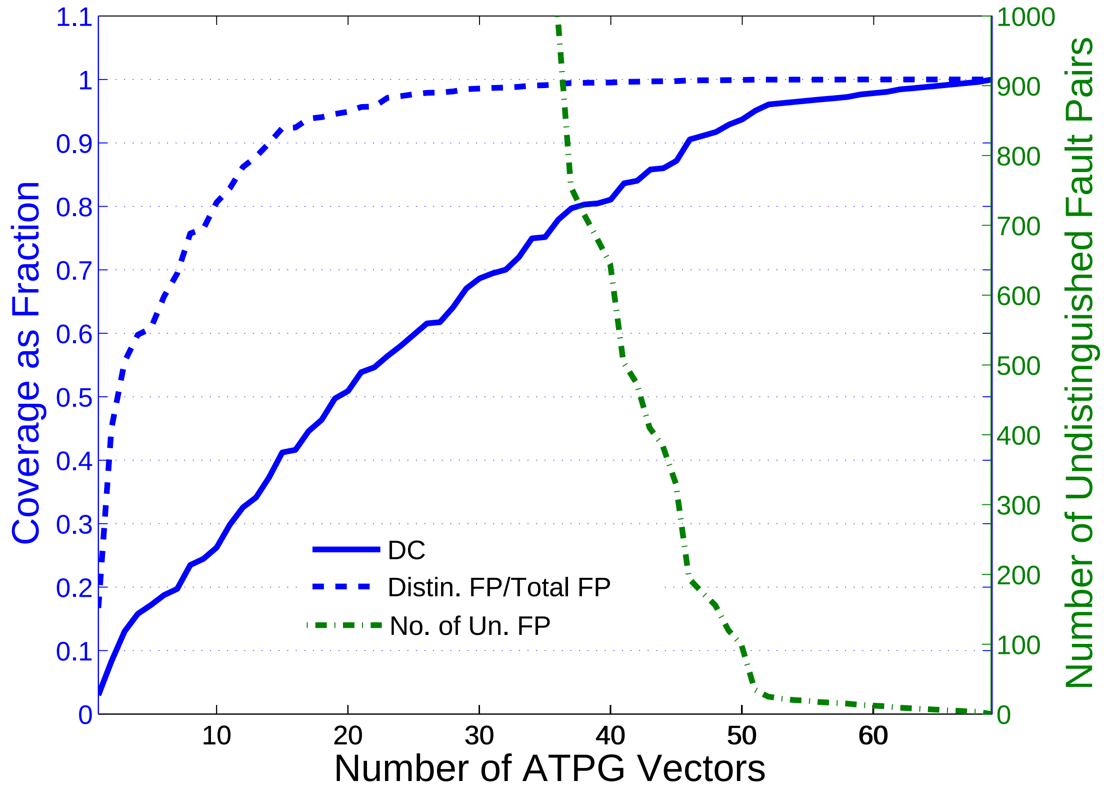


Figure 4.11: *DC* example of c432.

4.5 Diagnostic Fault Simulation and DATPG System

To construct a complete DATPG system a fault simulator is a necessity. We need an efficient algorithm to find undistinguished fault pairs for the ATPG to target. Once a test is generated, all remaining faults are simulated to get fault groups and *DC* is updated. The process stops when *DC* reaches a set value or an abort/timing limit is reached. A flow chart of the DATPG system is shown in Figure 4.12.

Key features of the new simulation algorithm are (1) it accepts fault detection data from any conventional fault simulator, thus benefiting from the efficiency of a mature program, (2) fault dropping is used to delete diagnosed faults from the list of faults as fault simulation progresses, and (3) for a given set of input vectors it provides fault coverage (FC), diagnostic coverage (DC), and necessary data for fault dictionary.

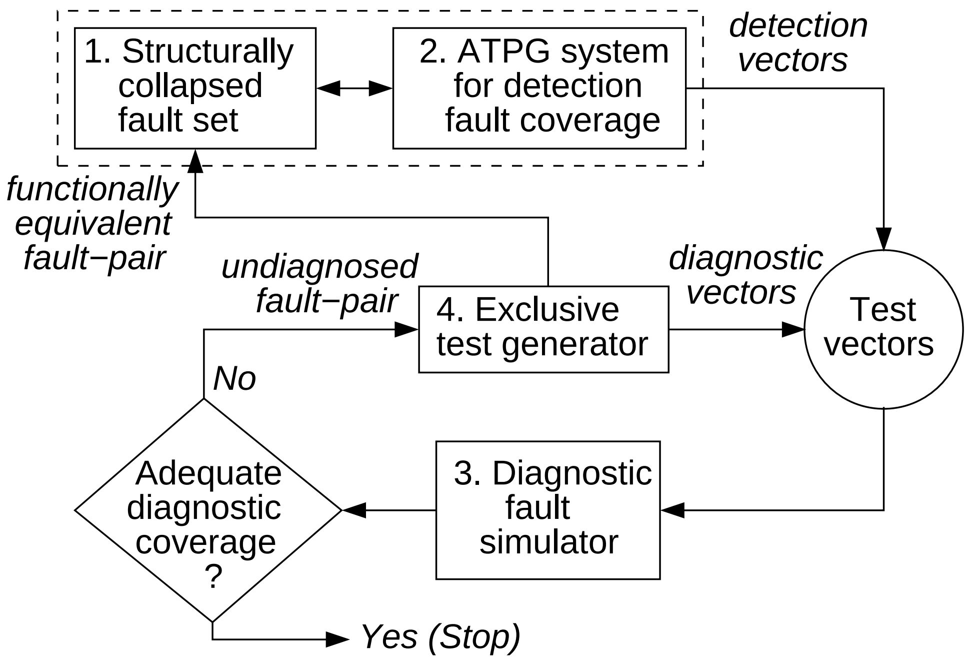
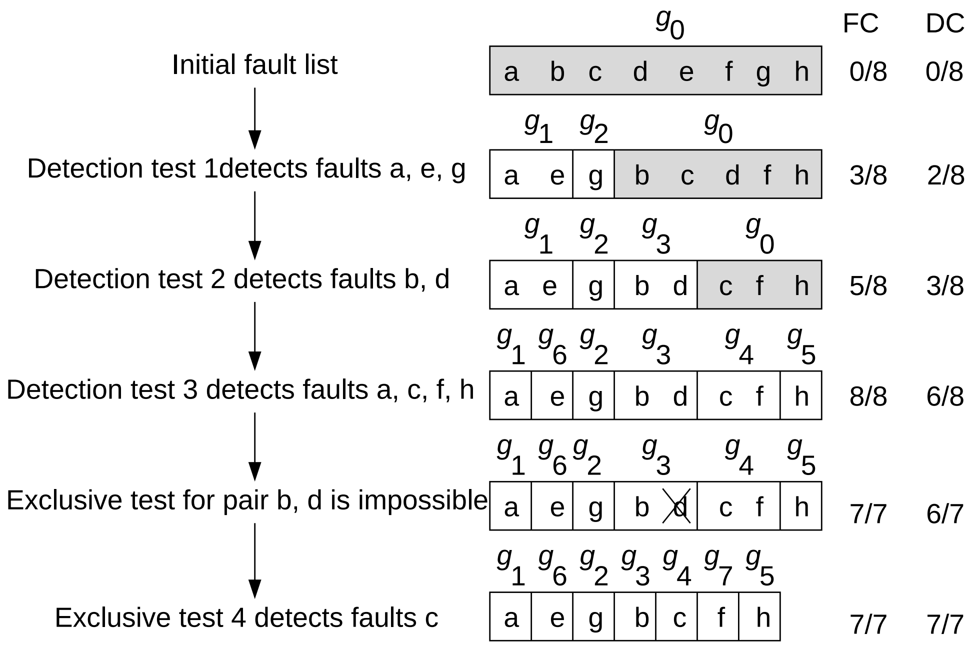

1. Structurally collapsed fault set
2. ATPG system for detection fault coverage
detection vectors
functionally equivalent fault−pair
undiagnosed fault−pair
4. Exclusive test generator
diagnostic vectors
Test vectors
No
Adequate diagnostic coverage ?
3. Diagnostic fault simulator
Yes (Stop)

Figure 4.12: Diagnostic test generation system.

We explain the simulation algorithm using a hypothetical fault simulation example given in Figure 4.13. Suppose a circuit has eight faults ($N = 8$), identified as a through h. Assume that the circuit has two outputs. The gray shading, which identifies the undetected fault group g_0, indicates that all faults are undetected in the initial fault list. Also, fault coverage (FC) and diagnostic coverage (DC) are both initially 0. We assume that there are three vectors generated in the detection phase and can detect all faults (blocks 1 and 2 in Figure 4.12), thus having 100% FC. Later in the simulation process more vectors will be generated by diagnostic ATPG to improve DC.

The diagnostic phase begins with the three vectors supplied to Block 3. The first vector is simulated for all eight faults using a conventional fault simulator. We use a full-response simulator that gives us the fault detection information for each primary output (PO). Suppose we find that the first vector detects a, e and g. Also, faults a and e are detected only on the first output and g is detected on both outputs. Thus, fault pairs (a, g) and (e, g) are distinguishable, while the pair (a, e) is not distinguishable. The result is shown in the second list in Figure 4.13. The fault list is partitioned into three groups. The first



g_0
FC
DC
Initial fault list
a b c d e f g h
0/8
0/8
g_1 g_2 g_0
Detection test 1detects faults a, e, g
a e g b c d f h
3/8
2/8
g_1 g_2 g_3 g_0
Detection test 2 detects faults b, d
a e g b d c f h
5/8
3/8
g_1 g_6 g_2 g_3 g_4 g_5
Detection test 3 detects faults a, c, f, h
a e g b d c f h
8/8
6/8
g_1 g_6 g_2 g_3 g_4 g_5
Exclusive test for pair b, d is impossible
a e g b d c f h
7/7
6/7
g_1 g_6 g_2 g_3 g_4 g_7 g_5
Exclusive test 4 detects faults c
a e g b c f h
7/7
7/7

Figure 4.13: Diagnostic fault simulation example.

two groups, g_1 and g_2, shown without shading, contain detected faults. Group g_0 now has 5 faults. Each group contains faults that are not distinguished from others within that group, but are distinguished from those in other groups. Counting detected faults, the fault coverage is 3/8 and counting detected fault groups, the diagnostic coverage is 2/8 [103].

Fault g, which is in a single fault group, is dropped from further simulation. This is similar to diagnostic fault dropping reported in other papers [95, 100]. Fault dropping greatly improves the efficiency. Previous dictionary based diagnosis procedures rely on fault simulation without fault dropping to generate a full response dictionary [11, 70, 73, 84, 90, 94], which is time and storage consuming because all faults have to be simulated with all test vectors.

Since fault g has been uniquely distinguished from all other faults, its distinguish-ability status will not change by other vectors. Note that pair-wise distinguish-ability provided by future vectors can only subdivide the groups and subdivision of a group with just one fault will be impossible. The fact that *faults can be dropped in diagnostic fault simulation* is not always recognized. However, fault dropping is possible here only because our interest is in diagnostic coverage and not in minimizing the vector set. Seven faults are now simulated for the second vector, which detects faults b and d. Suppose, b and d are detected at the same set of outputs and hence are placed within same partition g_3. Thus, $FC = 5/8$ and $DC = 3/8$. No new fault can be dropped at this stage.

Vector 3 detects faults a, c, f and h increasing the fault coverage to 100%. Suppose c and f are detected at the same set of outputs and so are placed together in group g_4. Detection at different outputs distinguishes h from these two and hence h is placed in a separate group g_5. Also, noting that this test distinguishes between a and e, group g_1 is split into g_1 and g_6. Now, $FC = 8/8 = 1.0$ and $DC = 6/8$. Faults in fault groups with single fault are dropped.

Having exhausted the detection vectors, we find that two pairs, (b, d) and (c, f), are not distinguished. We supply target fault pair (b, d) to Block 4 in the ATPG system of Figure 4.12. Suppose we find that an exclusive test is impossible thus indicating that the two faults are equivalent. We remove one of these faults, say d, from g_3 and from the fault list as well. This does not change fault coverage since $FC = 7/7$, but improves the diagnostic coverage to $DC = 6/7$. All faults except c and f are now dropped from further simulation.

The only remaining fault pair (c, f) is targeted and an exclusive test is found. Suppose fault f is detected by this vector but c is not detected Thus, g_4 is partitioned to create group g_7 with fault f. The new partitioning has just one fault per group, $FC = 7/7$, and $DC = 7/7$.

Figure 4.12 shows the flowchart of a diagnostic ATPG system implemented in the Python programming language [92]. Main functions are Blocks 1 through 4. Blocks 1 and 2 form a conventional detection coverage ATPG system. In our system, these functions are provided by Atalanta [52] and Hope [53], acquired from Virginia Tech. Note that any ATPG tools can be used. Block 4 is an exclusive test generator program that implements the core algorithm of Sections 3 and 5. Internally, it also uses Atalanta for detecting a fault on line y in the

ATPG circuit constructed for the given fault pair (Figure 4.9). Block 3 is a diagnostic fault simulator described before.

4.6 Dictionary Construction and Dictionary-less Diagnosis

A fault dictionary is necessary in conventional cause-effect diagnosis, though generated exclusive tests can also be used with effect-cause diagnosis. It facilitates faster diagnosis by comparing the observed behaviors with pre-computed signatures in the dictionary [91, 82, 83]. Based on the discussion in Chapter 3, our diagnostic fault simulator will generate an indexed full response dictionary on the fly. Only faults that are not dropped are simulated and signatures for the current test are stored. Signatures for previous tests are discarded, thus the memory required for diagnostic simulation is kept minimal. This scheme also provides a way to perform diagnosis without a pre-computed dictionary, which can grow impractically large for industrial designs. And it can help cutting down the cost of more and more expensive ATEs (Automatic Test Equipment).

Let's use Figure 4.13 in the previous section as an example. The dictionary shown in Table 4.2 is generated based on Figure 4.13 during simulation. Among the entries, X means the fault is already dropped and not simulated, 0 stands for pass (same as fault-free response), and 1 stands for fail. To reduce the dictionary size we assign integers to index different output responses. In this example,"1", and "11" are indexed with "1" and "2" for t1 as shown in Table 4.3. "01" is indexed with "1" for t2 and so on. As discussed in section 3.2 the largest index needed is usually much smaller than the highest number of faults detected by any test vector. We can see that even for the small example there is obvious reduction by comparing Table 4.2 and Table 4.3. For larger ISCAS'85 benchmark circuits the reduction can be over an order of magnitude. During simulation only one column of Table 4.3 is stored in memory. After simulating t1, we can discard the results because faults are already grouped for t1. To better illustrate this we can look at Figure 4.13. After t1 (a, e) are already put into one group, meaning they are distinguished from all other groups based

Table 4.2: Full response dictionary for Figure 4.13.

Faults	Signatures			
	t1	t2	t3	t4
a	10	00	10	X
b,d	00	01	00	X
c	00	00	01	00
e	10	00	00	X
f	00	00	01	11
g	11	X	X	X
h	00	00	10	X

Table 4.3: Indexed dictionary for Figure 4.13.

Faults	Signatures			
	t1	t2	t3	t4
a	1	0	1	X
b,d	0	1	0	X
c	0	0	2	0
e	1	0	0	X
f	0	0	2	1
g	2	X	X	X
h	0	0	1	X

on signatures of t1. In subsequent simulations only faults from same groups are targeted for "sub-division", thus previous simulation results are not needed.

Another application of this diagnostic simulator is dictionary-less cause-effect diagnosis. Still using Figure 4.13 as an example, suppose a failing signature of "00 00 01 11" is observed during testing. We simulate all faults for t1, and faults (a, e, g) are dropped. (b, d, c, f, h) are simulated for t2 and (b, d) are dropped. After t3 only (c, f) are left as possible candidates. And finally, with t4 c is eliminated. We get f as the one fault that is most likely to cause the failure. Physical inspection may be carried out to get final conclusion. One problem with this procedure is that there may be no match between observed signatures and simulated ones; more investigation is needed. Some sophisticated matching algorithms are proposed in [49, 51].

There is a limitation on our fault simulation algorithm. Because of fault dropping in our simulator there will be "X" in the generated dictionary. This limits the use of the fault dictionary to single stuck-at faults. For a real defect the faulty responses may have no match in the dictionary. To solve this problem we introduce a heuristic matching algorithm:

$$FC = \frac{Hamming\ Distance}{O(V - X)} \leq Threshold\ Value \tag{4.17}$$

Here hamming distance is calculated from observed response to the stored syndromes, ignoring "X"s. O is the number of primary outputs, V is number of vectors, and X is number of "X"s for a fault in the dictionary. If the calculated result is smaller than a given threshold, the corresponding fault will be added to a candidate list. Then fault simulation without fault dropping will be performed on this list to obtain additional information to further narrow down upon a fault candidate.

With the proposed exclusive test generation model, diagnostic coverage metric, and fault simulation scheme we keep the complexity of DATPG same with conventional ATPG, meaning that no additional effort is needed to extend the ability of fault testing tools for fault diagnosis.

4.7 Experimental Results

In this section we use ISCAS'85 [19, 36] combinational benchmark circuits to test our DATPG system.

4.7.1 Equivalence and Aborted Faults

In the ATPG system of Figure 4.12 when a single fault ATPG program is run it can produce three possible outcomes, (1) test found, (2) no test possible, or (3) run aborted due to CPU time or search limit in the program. In (1) a new detection or exclusive test is found. In (2), if it is detection test then the fault is redundant and is removed from the fault list. If it is an exclusive test then the two faults are equivalent (perhaps functionally) and either one is removed from the fault list. In (3), for detection phase the fault remains in the set g_0

and results in less than 100% FC and DC. For an aborted exclusive test the target fault pair remains indistinguishable, causing reduced DC and worse diagnostic resolution (DR) [6]. Hartanto et al. [37] introduced local circuit transformations and structural analysis to identify equivalences. Amyeen et al. [13] presented techniques based on implication of faulty values and evaluation of faulty functions in cones of dominator gates of fault pairs. The efficiency of our Diagnostic ATPG system will benefit from these methods of equivalence checking. We discuss this aspect in the next section.

4.7.2 Results

We used the ATPG system of Figure 4.12. Internally, it employs the ATPG program Atalanta [52] and fault simulator Hope [53]. The circuit modeling for exclusive test and fault grouping for diagnostic fault simulation are implemented in the Python language [92]. The system runs on a PC based on Intel Core-2 duo 2.66GHz processor with 3GB memory. The results for c432 were as follows:

- Fault detection phase:

  Number of structurally collapsed faults: 524

  Number of detection vectors generated: 51

  Faults: detected 520, aborted 3, redundant 1

  Fault coverage, FC: 99.43%

- Diagnostic phase:

  Initial DC of 51 vectors: 91.985%

  Number of exclusive tests generated: 18

  Number of undistinguished groups: 0

  Largest size of undistinguished group: 1

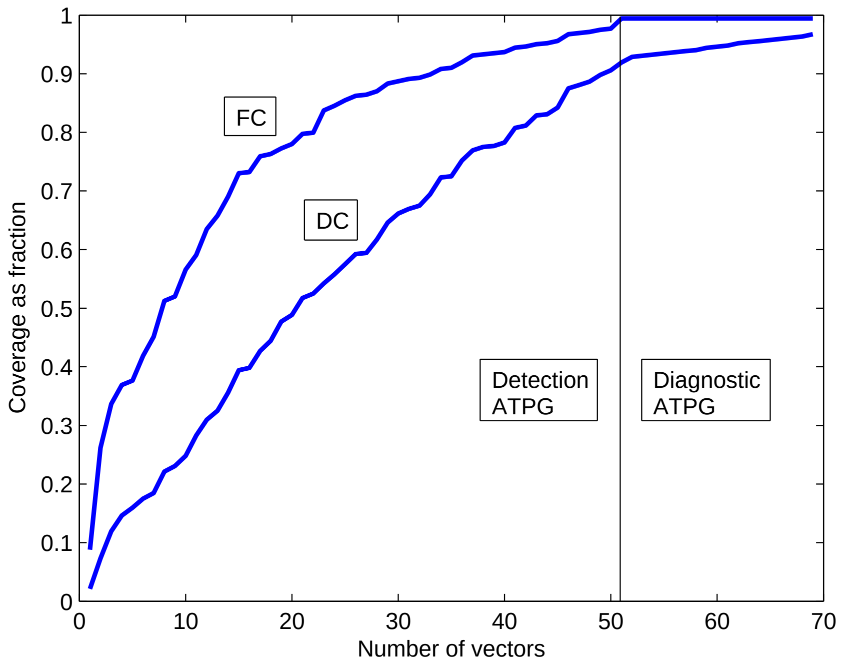


Figure 4.14: Diagnostic fault simulation of c432 for 69 algorithmic vectors. FC: fault coverage, DC: diagnostic coverage.

Final diagnostic coverage DC: 100%

Fault coverage (FC) and diagnostic coverage (DC) as functions of number of vectors are shown in Figure 4.14. First 51 vectors were generated in the fault detection phase, which identified only one of the known four redundant faults in this circuit. Diagnostic fault simulation computed the diagnostic coverage of 51 vectors as 91.985%. The diagnostic phase produced 18 exclusive tests while identifying 13 equivalent fault pairs. Diagnostic coverage of combined 69 vectors is 100%. No group has more than 1 fault. We also simulated a set of 69 random vectors and their coverages are shown in Figure 4.15. As expected, both fault coverage and diagnostic coverage are lower than those for algorithmic vectors. Results for ISCAS'85 circuits are shown in Table 4.4 and 4.5.

Table 4.5 indicates a dropping DC as circuit size increases. Once again, this is expected because of larger numbers of aborted pairs. Notice 740 aborted pairs for c1355. This circuit is functionally equivalent to c499, which has a large number of XOR gates. In c1355, each

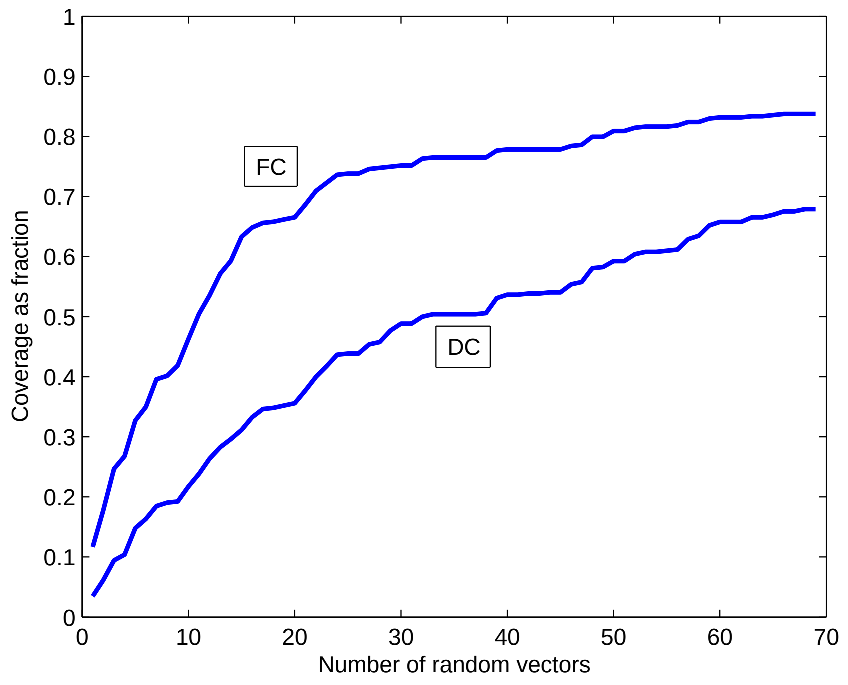


Figure 4.15: Diagnostic fault simulation of c432 for 69 random vectors. FC: fault coverage, DC: diagnostic coverage.

XOR gate is expanded as four NAND gates. This implementation of XOR function is known to have several functionally equivalent faults [9]. Its structurally collapsed set of 1,574 faults reduces to 950 faults when functional collapsing is used. If we use the set of 950 faults, the same $85 + 2 = 87$ vectors of Table 4.5 will show a significantly higher DC. Thus, the advantage of functional fault collapsing, though only marginal in detection ATPG, can be significant in diagnostic test generation.

Similarly, the size of the undiagnosed fault group tends to increase for larger circuits. It is 11 for c2670. This is related to the lower DC, whose reciprocal is the diagnostic resolution (DR). $DR > 1$ indicates poor diagnosis; the ideal resolution $DR = 1$ requires that each undistinguished fault group has no more than one fault.

The above discussion points to the need for improved fault collapsing and efficient redundancy identification algorithms. Much work on these has been reported [13, 37], which we are exploring for suitable approaches. Interestingly, we find that most redundancies or

Table 4.4: Experimental results for ISCAS'85 benchmark circuits (detection phase).

Circuit	No. of faults	Detection test generation			
		Det. tests	*FC* %	*DC* %	CPU s*
c17	22	7	100.0	95.45	0.031
c432	524	51	99.24	91.99	0.032
c499	758	53	100.0	97.36	0.032
c880	942	60	100.0	92.57	0.047
c1355	1574	85	100.0	58.90	0.046
c1908	1879	114	99.89	84.73	0.047
c2670	2747	107	98.84	79.10	0.110
c3540	3428	145	100.0	85.18	0.125
c6288	7744	29	99.56	85.32	0.220
c7552	7550	209	98.25	85.98	0.390

*Intel Core 2 Duo 2.66GHz, 3GB RAM.

Table 4.5: Experimental results for ISCAS'85 benchmark circuits (exclusive phase).

Circuit	Diagnostic test generation							
	Excl. tests	Abort pairs	Equ. pairs	Max. grp.	*DC* %	*DC*** %	CPU s	CPU s***
c17	1	0	0	1	100.0	100.0	0.33	0.030
c432	18	0	13	1	100.0	100.0	1.75	0.031
c499	0	12	12	1	98.40	100.0	0.39	0.031
c880	10	55	55	1	94.16	100.0	2.77	0.051
c1355	2	740	740	1	59.38	100.0	26.06	0.131
c1908	20	300	277	8	86.46	98.78	10.84	0.066
c2670	43	494	466	11	86.42	98.94	26.70	0.336
c3540	29	541	486	8	89.69	97.17	22.03	0.420
c6288	108	842	977	3	86.87	99.52	27.15	7.599
c7552	87	904	1091	7	86.85	99.35	26.36	2.181

**Equivalent pairs identified and *DC* after applying fault implication and dominator evaluation [13].

***CPU time excluding the time to rebuild data structure, which is avoidable through efficient implementation.

functional fault equivalences are localized within relatively small sub-circuits irrespective of how large the entire circuit is. Similar results are also reported in [37]. After applying equivalence checking, the DCs are greatly increased, as shown in table 4.5, column 7. Column 6 shows the *DC* before redundancy checking.

A comparison of two CPU time columns (column 6 and 8) of Table 4.4 and Table 4.5 shows that diagnostic ATPG takes significantly longer. This is because our implementation uses an existing ATPG program without changes. For detection ATPG, the circuit data structure is built once and then used repeatedly by every vector generation run. Even though the data structure building time is higher than that of a single vector generation, the total CPU time is dominated by the combined vector generation times. However, in diagnostic ATPG we repeatedly modify the netlist, and the ATPG program must rebuild the data structure prior to each vector generation run. An improved program will incrementally update the data structure instead of rebuilding it. That will bring the diagnostic ATPG time (column 8 of Table 4.5) in line with the detection ATPG time (column 6 of Table 4.4). For diagnostic fault simulation, the CPU time is linearly dependent on each of the three variables, namely, number of gates, number of faults and number of vectors. The CPU times in Table 4.5 include the time of the conventional fault simulator Hope [53] and that of our Python program that partitions the fault list and computes DC. The overall increase in run times with increasing circuit size for diagnostic simulation shown in Table 4.5 is between $O(gates^2)$ and $O(gates^3)$, which is no different from what has been reported for conventional fault simulation [15, 21].

Table 4.6 shows a comparison between the results obtained from our DATPG system when Atalanta [52] or a commercial tool Fastscan [63] is used. These results are for the ISCAS'85 benchmark circuit c7552. The data from Atalanta is taken from Tables 4.4 and 4.5. In the detection phase, Fastscan produced almost half as many vectors, 116 versus 209 from Atalanta. As may be expected from a commercial ATPG tool, Fastscan has a more comprehensive vector minimization built in strategy. It also has highly efficient search algorithms and is able to identify more redundant faults. Hence, a higher fault coverage, $FC = 99.94\%$ versus 98.25% for Atalanta. The corresponding diagnostic coverages for the detection vectors were 85.98% for Atalanta and 86.95% for Fastscan.

Table 4.6: Diagnostic test generation results for ISCAS'85 benchmark circuit c7552.

Test system	Detection phase			Diagnostic phase			
	Detect. vectors	*FC* %	*DC* %	Exclu. vectors	Total vectors	Aborted fault pairs	Final *DC* %
DATPG (Atalanta)	209	98.25	85.98	87	296	904	86.85 (99.35)
DATPG (Fastscan)	116	99.94	86.95	44	160	9	99.80
N-detect ILP [83]	122	∼98	85.47	31	153	—	86.88

In case of Atalanta, exclusive tests were generated for all undiagnosed fault pairs before using diagnostic fault simulation. In contrast, for Fastscan every exclusive test was simulated for diagnostic coverage before generating the next exclusive test. This strategy produced fewer exclusive tests, 44 versus 87 for Atalanta. Perhaps compaction of the Atalanta vectors will close this gap. More noticeable is number 904 of aborted fault pairs for the exclusive test generation by Atalanta. With Fastscan, all but 9 fault pairs either produced an exclusive vector or were found to be equivalent through the comprehensive search build into Fastscan. This resulted in a high DC of 99.8%. Atalanta-based DATPG produced DC = 86.85%, which was then updated to 99.35% shown in parenthesis in Table 4.6 by additional fault equivalence analysis of 904 fault pairs. The final fault coverage (FC) of the total set of vectors remained the same as obtained by the initial detection vectors (column 3).

We could not exactly compare the run times of Atalanta and Fastscan because they ran on different computing platforms. However, their normalized CPU times may not differ much. The result of DATPG with Atalanta required additional equivalence pair identification consuming extra CPU time. One difference between the two cases is the diagnostic ATPG strategies. The one adopted with Fastscan, following each exclusive vector by an immediate diagnostic fault simulation, proved to be better and is recommended.

The last row in Table 4.6 gives a result reported in previous publications [80, 81, 82, 83]. In that work the authors minimize test sets without reducing DC. They start with an N-detect test set and use a primal-dual integer linear programming method to compact the test set while also applying additional heuristics. For this result, they started with 4924 N-detect vectors for c7552 circuit and minimized them in two steps. The fault coverage (FC)

of these vectors was approximately 98%. First, in a detection step, they obtained a minimal test set of 122 vectors to cover all faults detectable by the given vectors. These vectors had $DC = 85.47\%$, computed as reciprocal of the reported diagnostic resolution $DR = 1.17$. Then, in a diagnostic step, from the remaining $4924 - 122 = 4802$ vectors, 31 were selected for preserving the diagnostic capability of the original vectors. These vectors boosted DC to 86.88% without affecting the fault coverage. This DC will most probably rise to similar levels as obtained the other two cases in Table 4.6 if exclusive tests were attempted for undiagnosed fault pairs or they were analyzed for equivalence. In comparison, the DATPG approach is more direct, based on generation of detection, and exclusive tests and does not require an ILP-type of minimization, which can be expensive.

4.8 Conclusion

The core algorithms for exclusive test generation and diagnostic fault simulation should find effective use in the test generation systems of the future. Key features of these algorithms are: (1) exclusive test generation is no more complex than test generation for a single fault, and (2) diagnostic fault simulation has similar complexity as conventional simulation with fault dropping. The new definition of diagnostic coverage (DC) is no more complex than the conventional fault detection coverage and it directly relates to the diagnostic resolution [6], which is the reciprocal of DC. These methods can be applied to other fault models [38, 54, 55] and to mixed fault models [99]. Future extensions of this work would be on generating compact diagnostic tests and organizing fault dictionaries of test syndromes. Because our fault simulation is done with fault dropping, the syndromes will contain 0, 1, and X (don't care). However, these don't cares do not reduce the diagnosability of a fault, although, reordering or compaction of vectors will be affected. This needs further investigation. Our results indicate that to obtain high DC for large circuits, efficient heuristics and algorithms for functional equivalence checking and redundancy identification are essential. The presented diagnostic ATPG system can also be applied to transition faults after some modification, since transition faults are actually based on stuck-at faults. This can be a future expansion of the ATPG system.

Chapter 5

Diagnostic Test Generation for Transition Faults

As discussed in Chapter 1 more and more failures of modern VLSI chips are timing related. For example hold and setup violations will cause incorrect values being captured in flip flops. If the clock speed is too fast or there is excessive delay along a certain path, it may cause a setup violation, meaning that data arrives too late. If a path is too short or a previous flip-flop cannot hold an output value long enough, it may cause hold violations. Usually many iterations are needed during place and route to fix all timing violations [64]. It will be very helpful to improve design methodology or product quality if we can diagnose a timing related fault.

Two types of delay fault models are introduced in Chapter 2. Since transition fault is very similar to stuck-at fault, in this chapter we propose an algorithm to extend the ability of our DATPG system to be able to target transition faults. We also discuss the potential to further extend its ability for other fault models, such as path delay fault [5], bridging fault, etc.

With a DATPG capable of targeting multiple fault models we can distinguish between different types of faults. This greatly improves the accuracy and application area of the proposed method.

5.1 Introduction

Chapter 4 describes a system for generating diagnostic tests using the single stuck-at fault model. Main ideas introduced there were a definition of diagnostic coverage and algorithms for diagnostic simulation and exclusive test [6] generation. In that work emphasis was placed on using the existing tools that were originally designed for fault detection only.

The work presented in this chapter provides a similar capability for the diagnosis of transition faults. A reader will find these extensions to be non-trivial. In this work we emphasize the use of existing tools and techniques to implement the new algorithms. The basic tools used are the simulation and test generation programs for detection of single stuck-at faults. Scan test environment is assumed in which both launch-off-capture (LOC) and launch-off-shift (LOS) types of tests can be conducted. Figure 2.8 of section 2.4 shows a basic scan design using multiplexers to choose from scan mode and function mode. The two testing schemes (LOC, LOS) will be presented in detail in section 5.6. It is well known that a transition fault test contains a pair of vectors applied to the combinational part of the circuit [20]. In scan testing, as the first of these vectors is scanned in it appears at the input of the combinational logic. The second vector is then either produced by clocking the circuit in the normal mode (known as launch-off-capture or LOC test [79]) or in the scan mode (launch-off-shift or LOS test [78]), following which the response is captured in scan register always in the normal mode and scanned out in the scan mode. In Figure 2.8, switching between scan mode and normal (functional) mode is controlled by a Scan Enable pin. Enhanced scan [27] has the capability of applying any vector pairs at the cost of extra hardware (flip-flops are doubled), which is not widely used.

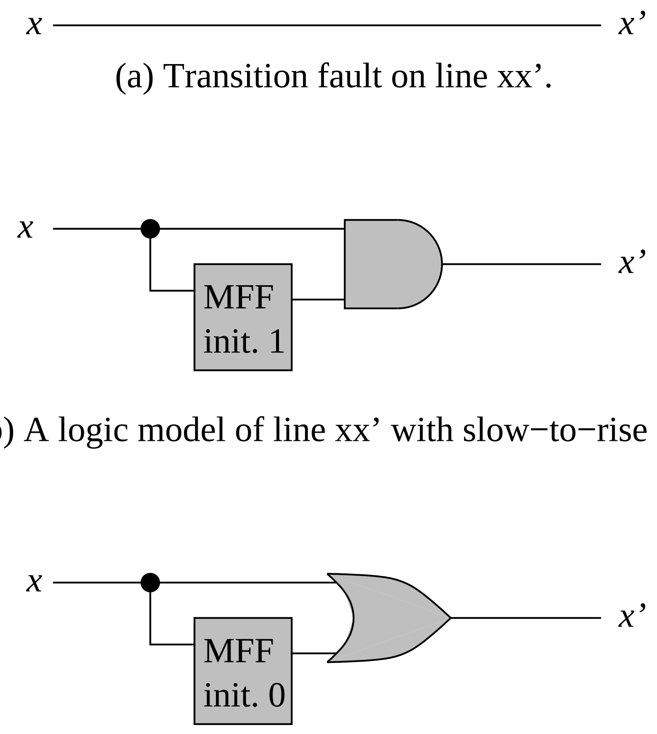
Our principal objective in this work is not just to test for transition faults, as is normally done, but to distinguish the exact fault that may have caused a failure. We continue to use the diagnostic coverage (DC) metric from the previous chapter. The new modeling techniques and algorithms for transition faults are more efficient than those published before [38]. Our implementation and results support the practicality of the presented approach.

5.2 Background

The usefulness of the transition fault model stems from the fact that modern VLSI devices must be tested for performance. Transition faults are not perfect and in fact may not represent many of the actual defects. Their acceptability, like that of stuck-at faults, is due to several practical reasons. To name a few, their number grows only linearly with circuit size, they require two-pattern tests that are essential for detecting delay and other non-classical faults, and the scan methodology can be adapted to test them.

Why is the diagnosis of transition faults important? The same technology advances that give us lower cost and higher performance make it necessary that we diagnose delay defects. Presently, we must rely on ad-hoc measures like at-speed testing, N-detect vectors, etc. N-detect tests have shown promising ability to detect/diagnose non-classical faults [59]. By increasing N, we expect that more faults, including some timing related faults, can be captured. But the number of test patterns also grows with increasing N, which leads to longer test time and/or higher cost. Methods have been proposed [46, 47, 82, 83] for compaction of N-detect vector sets. These require computational effort. The present work [104] is aimed at providing a similar diagnostic capability for transition faults as is available for stuck-at faults [6, 35, 100, 102] without increasing the ATPG effort, while also keeping test pattern count down.

A previous attempt [38] at generating diagnostic tests for transition faults used a complex model requiring up to four copies of the combinational circuit. This increases the complexity of the ATPG, which is already known to be an NP-complete problem [20]. Besides, that method is demonstrated to work only for LOC tests. The ATPG model presented here requires only one copy of the circuit, although the model is mildly sequential. The sequential behavior comes from one or two modeling flip-flops that are added to the circuit. These flip-flops are pre-initialized and hence do not increase the ATPG complexity. The new procedure is equally suitable for both LOC and LOS tests. It is also more flexible since both sequential and combinational ATPG can be used. For combinational ATPG the CUT needs to be unrolled into two time frames. It is still more efficient compared to [38] because only two, and not four, copies of the CUT are required. SAT based diagnostic test generation procedures have been proposed [16] for the transition fault model. A SAT formulation may be too complex for large designs.



(a) Transition fault on line xx'.

(b) A logic model of line xx' with slow−to−rise fault.

(c) A logic model of line xx' with slow−to−fall fault.

Figure 5.1: Modeling a faulty circuit with a single transition fault on line xx'.

5.3 Modeling a Transition Fault

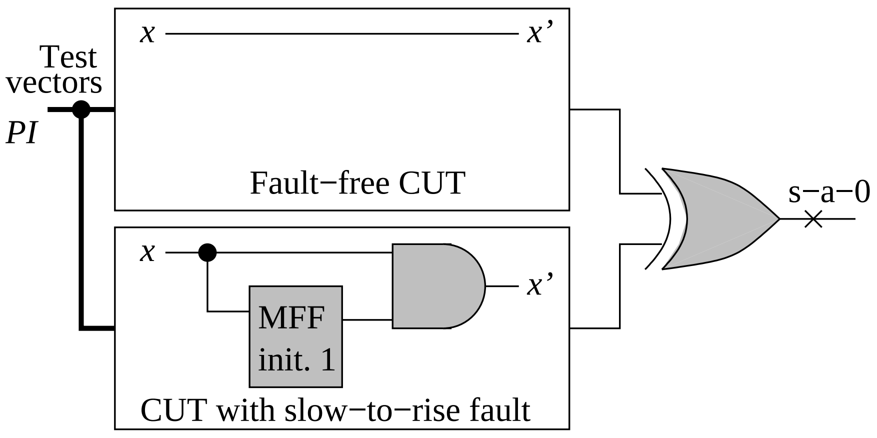
Because we wish to use the existing methods that are based on the logic description of the fault condition, we will use a synchronous model for the transition fault. Figure 5.1 shows a method of modeling a single slow-to-rise or slow-to-fall transition fault on a line xx' in the combinational logic with a synchronous sequential sub-circuit. The shaded elements are inserted for modeling the fault and do not belong to the actual circuit. The modeling flip-flop (MFF) is initialized to the value shown. Consider the slow-to-rise fault in Figure 5.1(b). Flip-flop initialization to 1 ensures that the output x' on the line will be the correct logic value on the first vector. Of the four two pattern sequences on x, 00, 01, 10 and 11, all except 01 will produce the correct output at x'. The sequence 01 at x will appear as 00 at x', correctly representing a slow-to-rise transition fault on line xx'. Figure 5.1(c) shows a similar model for a slow-to-fall transition fault on line xx'.

5.4 An ATPG Model

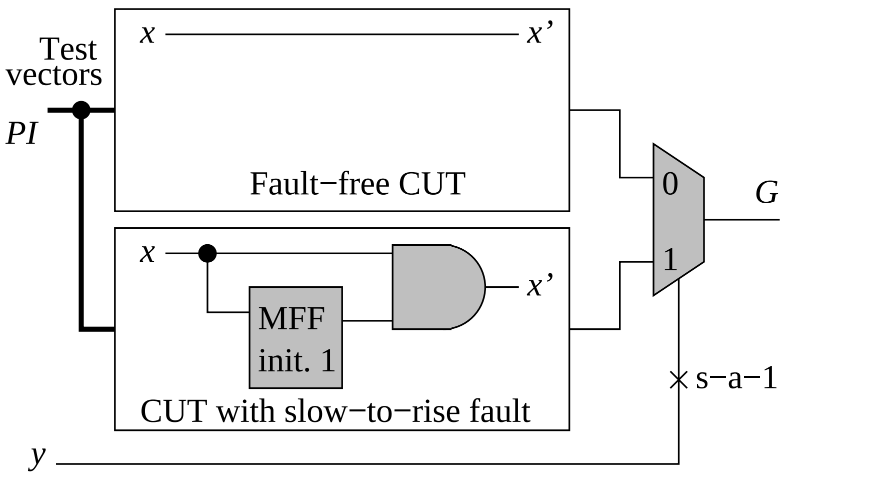
An ATPG (automatic test pattern generation) model is a netlist of the circuit under test (CUT), modified to represent the target fault as a stuck-at fault. The modification amounts to insertion of a few logic elements for modeling only. For a transition fault, we construct a single stuck-at fault ATPG model as shown in Figure 5.2. The ATPG model of Figure 5.2(a) gives the conventional Boolean satisfiability formulation. Note that a 1 output from the EXOR gate cannot be obtained by a single vector. Because the modeling flip-flop MFF is initialized to 1, initially, $x' = x$. To produce a different output from the faulty circuit, the first vector must set $x = 0$ and then a second vector should set $x = 1$, besides sensitizing a path from x' to the primary output (PO) or an observable point (e.g. a scan flip flop).

The ATPG model of Figure 5.2(b) can be used in the same way [102]. Any test sequence for either s-a-0 or s-a-1 fault on y must produce different outputs from the fault-free and faulty circuits. The advantage of this model is that it can be simplified to use a single copy of the circuit, which is an improvement over the previous work [38]. The analysis that leads to the ATPG model of Figure 5.3 is the same as given in Chapter 4. There a single-copy ATPG model was obtained for finding an exclusive test for a pair of stuck-at faults. Thus, the fault-free CUT in Figures 5.2 (a) and (b) was replaced by the CUT containing one of the faults.

The main idea that allows us to collapse the two copies of the circuit in Figure 5.2(b) into a single copy is the realization that the two circuits are almost identical. The only difference is at the faulty line. It can be shown [102] that the multiplexer at PO can be moved to the fault site. The procedure is as follows: Suppose a transition fault is to be detected on a signal interconnect from x (source) to x' (destination). In a single copy of the circuit, the source signal x is made to fan out as two signals $x1$ and $x2$. Fanout $x1$ is left as fault-free signal. The other fanout $x2$ is modified according to Figure 5.1 to produce the faulty value. These two signals $x1$ and $x2$ are applied to the two data inputs of a multiplexer



(a) An ATPG model: test for output s−a−0 fault detects the slow−to−rise fault on xx'.



(b) An alternative ATPG model: test for stuck−at fault on y detects the slow−to−rise fault on xx'.

Figure 5.2: ATPG model with a stuck-at fault for detecting a slow-to-rise fault on line xx'.

whose output x' now feeds into all destinations of the original x', and whose control input is a new PI y. The target fault now is any stuck-at fault (s-a-0 or s-a-1) on y. Any test for this target fault must produce different values at fault-free $x1$ and the faulty $x2$ while propagating the value of x' to a PO, and hence must detect the fault modeled by $x2$. This ATPG model for a slow-to-rise fault on xx' is shown in Figure 5.3. Any test for y s-a-0 or for y s-a-1 in the ATPG model of Figure 5.3 will always contain two vectors. The model for a slow-to-fall transition fault is obtained by replacing the AND gate by an OR gate and changing the initialization of the flip-flop to 0, as shown in Figure 5.1(c). The gate and

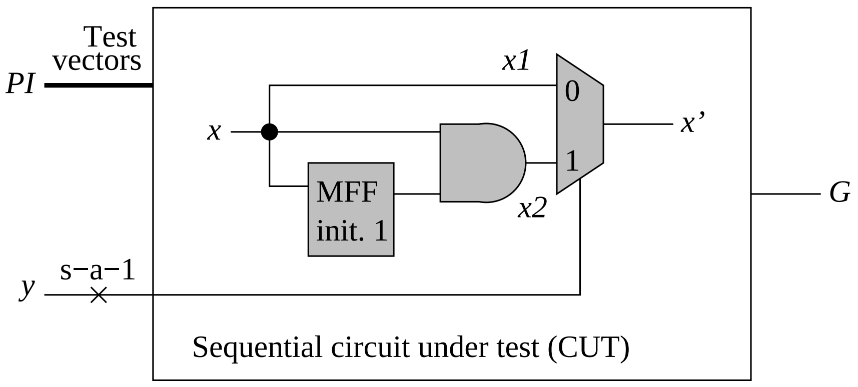


Figure 5.3: A single circuit copy ATPG model in which a test for a stuck-at fault on y detects the slow-to-rise fault on line xx'.

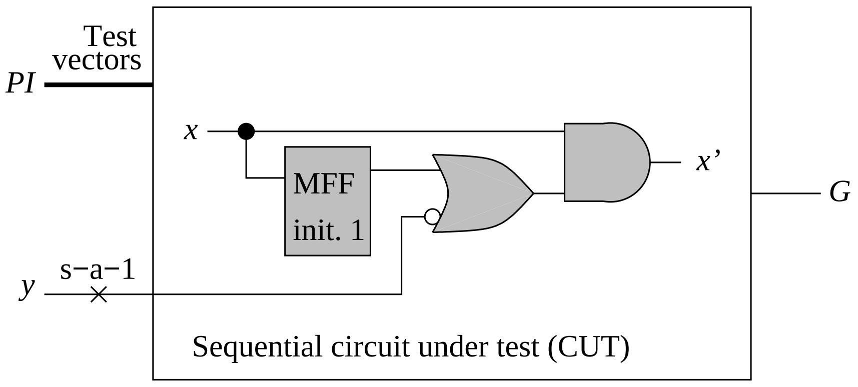
multiplexer combination can be further simplified to an equivalent ATPG model given in Figure 5.4, which shows the ATPG models for both slow-to-rise and slow-to-fall transition faults.

5.5 Combinational and Scan Circuits

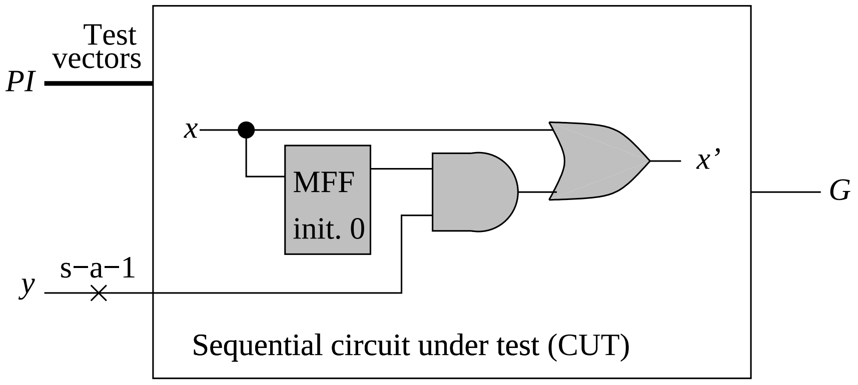
The above fault modeling procedure is valid for both combinational and scanned sequential circuits. For a combinational circuit under test (CUT), the modeling flip-flop (MFF) serves two purposes. First, it requires a two-vector test. Second, the initial state of the flip-flop makes it impossible to activate the fault effect at x' in the first vector. This model can be used to generate a two-vector test either by a sequential ATPG program or by a combinational ATPG program applied to a two time-frame expansion of the circuit.

For a scanned sequential circuit under test (CUT) the ATPG models of the previous section will also generate two-vector tests. The vectors can be generated either by a scan ATPG program in the partial scan mode to accommodate the modeling flip-flop (MFF) or by a combinational ATPG program. The second vector is generated either as a launch-off-capture (LOC) sequence or as a launch-off-shift (LOS) sequence.

Figure 5.5 shows the two time-frame circuit for a combinational ATPG program.



(a) Slow−to−rise transition fault on line xx'.



(b) Slow−to−fall transition fault on line xx'.

Figure 5.4: Simplified single circuit copy ATPG models in which a test for a stuck-at fault on y detects a transition fault on line xx'.

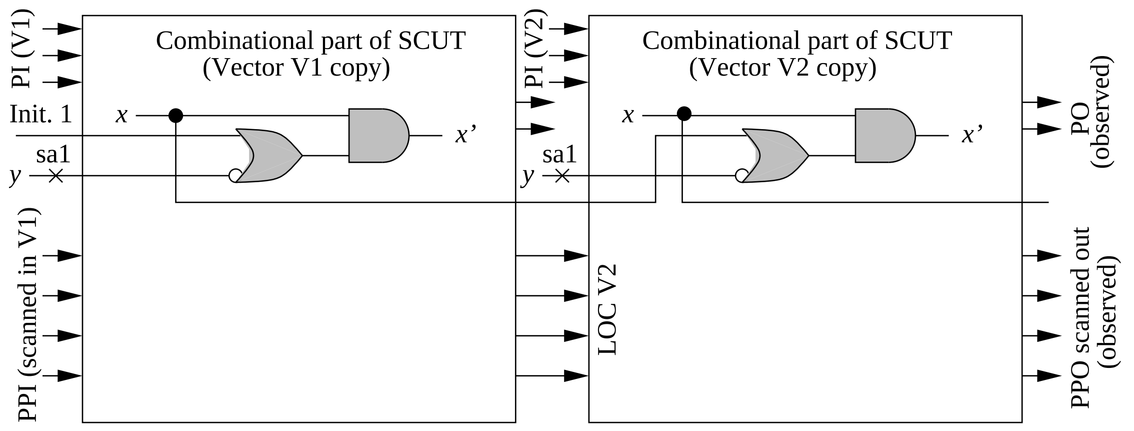
5.6 Scan Test Generation

Consider a sequential circuit to be tested via scan. An ATPG tool like Mentor's Fastscan [61] will generate scan sequences for all stuck-at faults in the combinational logic of the circuit. Fastscan can also generate two-pattern scan sequences for all transition faults in the combinational logic. At the user's option, it generates tests for application in either LOC or LOS mode. Fastscan allows test generation in the partial scan mode as well, provided the number of non-scan flip-flops is small. That capability is useful for the ATPG model of Figure 5.3 which requires a single non-scan flip-flop.

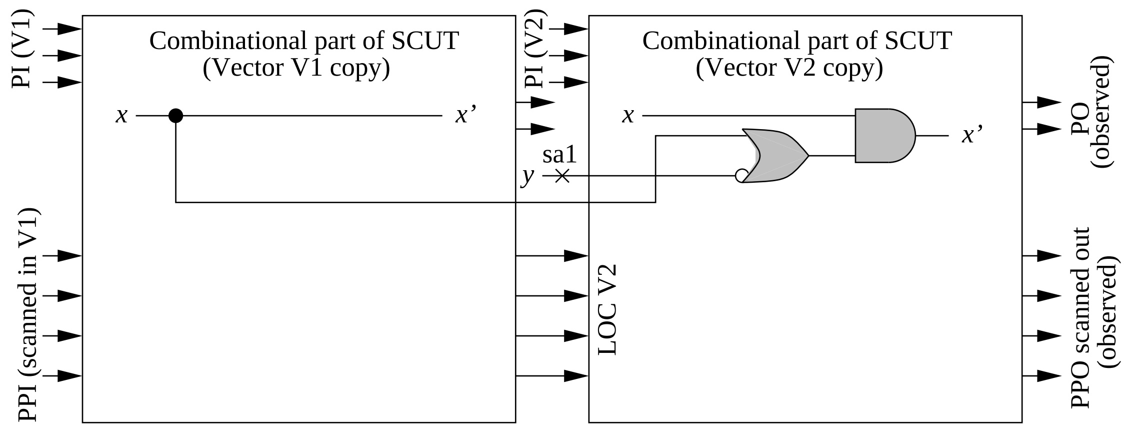
A sequential circuit is first modified for scan testing using Mentor's DFTAdvisor [62]. The modeling flip-flops are treated as non-scan pre-initialized flip-flops. Thus, scan-in, scan-out and scan-enable (SE) signals are added to the circuit. The ATPG program Fastscan [61] then generates a test for a single stuck-at fault. This test consists of a scan-in sequence and two primary input vectors. The application sequence of this test is: (1) Set SE = 1 (scan mode) and apply the scan-in sequence, (2) Apply first primary input vector, (3) Set SE = 0 (normal mode), apply second primary input vector and clock the circuit at the same time to launch the second vector, (4) Clock the circuit again with functional clock, which is faster than shift clock, to capture responses in scan flip-flops, (5) Set SE = 1 and scan out the captured results as well as scan in the next vector with shift clock. This sequence will test the fault in the LOC mode. For LOS mode, we hold SE = 1 (scan mode) for one cycle, which Fastscan allows. In that case, the ATPG assumes that the circuit will be clocked in the scan mode for a specified number of clock periods (one, in this case) while new primary inputs are applied. This test will then detect the modeled transition fault in the LOS mode. One disadvantage of LOS mode is that it requires a global scan-enable signal. All scan flip-flops need to change modes (normal, scan) simultaneously. And it is very difficult to implement a scan-enable signal with low skew. Generally LOS can achieve better coverage than LOC test, but not widely used.

The above ATPG model allows test generation for transition faults using the conventional stuck-at fault test generation tools. Fastscan, however, can directly generate tests as well as simulate them for transition faults. In our experiments, we use that capability of Fastscan. The ATPG models of Figures 5.3 and 5.4 will be used in the next section for generating exclusive tests for transition faults.

These models are especially useful when we generate tests using a combinational ATPG program. Both types of two-vector $(V1, V2)$ tests, namely, LOC and LOS, can be generated. Figure 5.5(a) shows a combinational circuit for LOC test of a slow-to-rise fault. It contains two copies of the combinational part of the sequential circuit. The fault is modeled using the



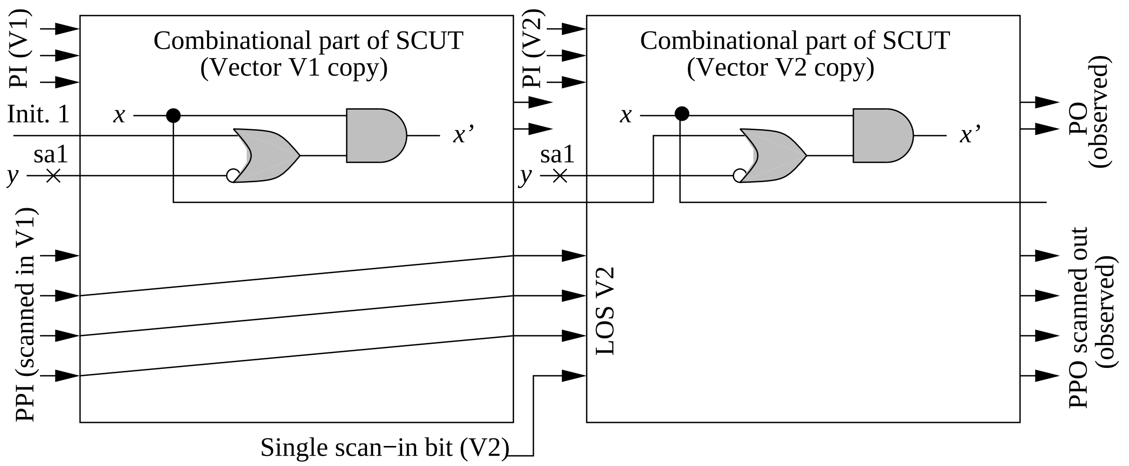
(a) Two time−frame combinational circuit for LOC test of slow−to−rise transition fault on line xx'



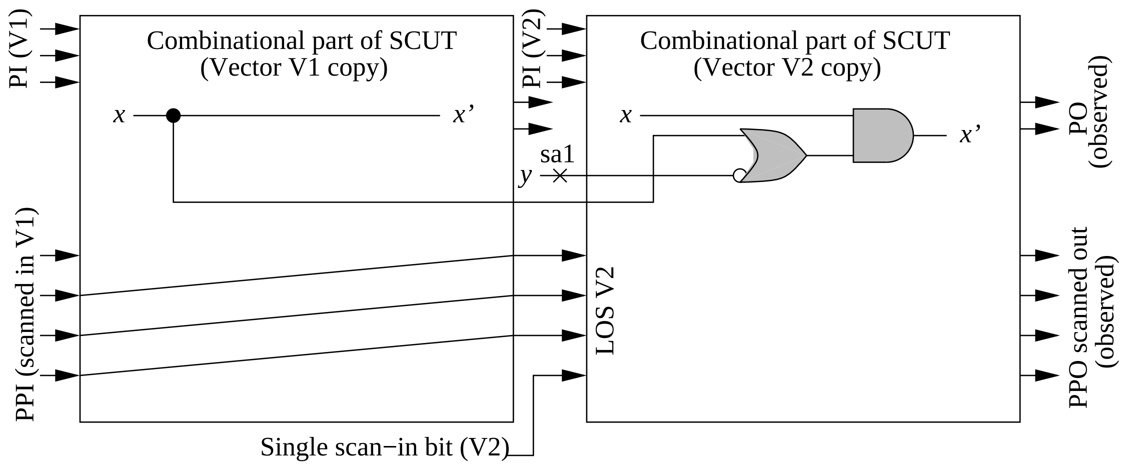
(b) Simplified combinational ATPG circuit; a test for y s−a−1 is a LOC test for slow−to−rise fault on xx'

Figure 5.5: Two time-frame circuit for a LOC transition test for slow-to-rise fault on line xx' by a combinational ATPG program targeting the fault y s-a-1.

construction of Figure 5.4(a). In the first time-frame the initial state, 1 (shown as init. 1), of the unscanned fault modeling flip-flop (MFF) is applied through an extra primary input (PI) fixed at 1. All scan flip-flops (SFF) are stripped off and replaced by pseudo primary inputs (PPI) and pseudo primary outputs (PPO). Vector $V1$ consists of the normal PI and PPI. Vector $V2$ consists of the PI of the second time-frame where the PPI are the PPO of the first time-frame. All outputs of the second time frame are observable, PO directly and PPO through scan-out. The circuit of Figure 5.5(a) has two faults. A closer examination, however, shows that it is impossible for the first y s-a-1 fault to cause any effect in the first



(a) Two time−frame combinational circuit for LOS test of slow−to−rise transition fault on line xx'



(b) Simplified combinational ATPG circuit; a test for y s−a−1 is a LOS test for slow−to−rise fault on xx'

Figure 5.6: Two time-frame circuit for a LOS transition test for slow-to-rise fault on line xx' by a combinational ATPG program targeting the fault y s-a-1.

time-frame due to the fixed "init. 1" input. Thus, the circuit can be simplified as shown in Figure 5.5(b) with a single stuck-at fault y s-a-1 for which any conventional combinational ATPG program can be used to obtain a test.

Figures 5.6(a) and (b) show two time-frame combinational circuit for a LOS test for a slow-to-rise transition fault. The basic difference from the LOC model of Figures 5.5(a) and (b) is in the way the PPI bits are obtained in the second time-frame. For LOC test these bits are obtained by a one-bit shift of the PPI bits of $V1$.

Similar combinational circuit models for LOC and LOS tests can be obtained for a slow-to-fall transition fault by using the equivalent circuit of Figure 5.4(b).

To generate exclusive test with 2 time-frame model we can use the stuck-at DATPG model described in Chapter 4. By inserting up to 3 primitive logic gates at the 2 new y inputs (one input for each transition fault), we can generate exclusive test for the transition fault pair with only 2 copies of the CUT.

5.7 Diagnostic Test Generation

The main contribution of previous sections is in modeling of a transition fault as a single stuck-at fault. The benefit of this model is that we can use the tools and techniques available for single stuck-at faults. We now illustrate the use of the following techniques discussed in Chapter 4 for transition faults:

1. Diagnostic coverage (DC) of tests that provides a quantitative measure for their ability to distinguish between any pair of faults.

2. Diagnostic fault simulator that determines DC for any given set of vectors and identifies undistinguished fault pairs. This diagnostic fault simulator internally uses any conventional single stuck-at fault simulator or any transition fault simulator.

3. Exclusive test generator that derives an exclusive test for a fault pair such that the two faults in the pair can be distinguished from each other. If an exclusive test is found to be impossible then the two faults are equivalent and one of them can be removed from the fault list to further collapse it. This exclusive test generator internally uses a conventional single stuck-at fault test generator.

4. A complete diagnostic test generation system that first generates the conventional tests for fault detection coverage, determines the DC of those tests, and then generates more vectors if necessary to enhance DC.

Table 5.1: Transition fault diagnostic test generation for ISCAS'89 [18] benchmark circuits. Circuits have full scan and tests are generated for application in LOC mode.

Circuit	No. of faults	Detection test generation					Diagnostic test generation			
		Detect tests	*FC* %	*DC* %	Undiag. flt. grp.	Lgst. grp.	Excl. tests	*DC* %	Undiag. flt. grp.	Lgst. grp.
s27	46	11	100.0	52.2	12	7	18	97.8	1	2
s298	482	44	79.9	62.4	62	5	34	70.1	39	4
s382	616	51	80.8	64.1	82	4	24	68.5	58	4
s1423	2364	102	92.9	79.3	280	5	106	84.1	182	5
s5378	6589	205	91.2	82.0	400	9	472	90.0	85	7
s9234	10416	377	92.8	75.8	1219	11	597	82.1	754	8
s13207	14600	480	89.1	70.0	1707	20	543	74.1	1392	11
s15850	17517	306	87.6	71.2	1961	9	486	74.3	1565	7
s35932	52988	75	99.0	88.3	3737	6	725	90.2	2867	4
s38417	47888	244	98.4	87.5	4090	9	1336	91.0	2883	8
s38584	56226	395	95.7	86.7	4042	8	1793	90.3	2440	7

Table 5.2: Comparison between previous work [38] and ours for s38584.

s38584	No. of faults	Detection test generation				Diagnostic test generation			
		Detect tests	*FC* %	*DC* %	Undiag. flt. grp.	Excl. tests	*DC* %	Undiag. flt. grp.	CPU s**
Previous	1000	2120	—	97.16*	—	583	97.42*	—	174649
This work	56226	395	95.7	86.7	4042	1793	90.3	2440	14841

*Fault pair coverage (number of distinguished pairs divided by total number of pairs)
**Hardware configuration is not reported in previous work
In our work: 2 x 1.0 GHz, 1885 MB RAM, x86 Linux

5.8 Results

The results of these procedures when applied to transition faults are shown in Table 5.1.

We used Fastscan [61] to generate fault detection tests for transition faults. Fastscan can generate transition fault tests for full-scan circuits in either of the two (LOC and LOS) modes. The results of Table 5.1 are for LOC mode only. The equivalent circuits of Figure 5.4 provide an alternative method. Here the target transition fault is represented as a single stuck-at fault. The modeling flip-flop (MFF) starts with a specified initial state and is not scanned. Thus, Fastscan generates a test for a single stuck-at fault y s-a-1 in the partial scan mode; all normal flip-flops of the circuit are scanned and the modeling flip-flop MFF is not scanned. All flip-flops, including MFF, are assumed to have the same clock. Because of the initial state of the unscanned MFF, the fault cannot be detected by the first vector, which merely initializes the circuit. The test consists of two combinational vectors, i.e., a scan-in sequence, followed by one clock in normal mode (LOC) or in scan mode (LOS), capture, and a scan-out sequence.

The second column of Table 5.1 lists the number of transition faults. Faults on the same fanout free interconnect and the input and output of a not gate are collapsed [63]. Also some of the redundant transition faults are identified during ATPG and they are removed. The third column lists the number of LOC tests. Note that Fastscan can perform test pattern compaction. Since in this work our focus is on the ATPG algorithm, we did not perform compaction on diagnostic test patterns. Each test consists of a scan-in, launch, capture and scan-out sequence. The detection fault coverage (FC) of transition faults is given in column 4. Reasons for less than 100% FC are (a) aborted ATPG runs, (b) LOS mode not used, and (c) redundancy or untestability not identified. Because Fastscan for transition faults operates in sequential mode, it often fails to identify redundancies. In our ongoing work we are developing a combinational two time-frame model mentioned in Section 5.6 to improve the fault efficiency. Based on observations made from several small ISCAS'89 circuits through detailed structural analysis we find that most aborted pairs are actually functionally equivalent. If all fault equivalences are identified, similar to fault efficiency, "diagnostic efficiency" would be much higher than diagnostic coverage. Note that FC can be considered as an upper bound on DC and fault efficiency is an upper bound on "diagnostic efficiency". The experimental results from [38] show that all targeted fault pairs are either distinguished or identified as equivalent pairs. Using 2-time-frame expansion we should get similar results. This needs further investigation.

Column 5 of Table 5.1 gives the diagnostic coverage (DC) obtained from diagnostic fault simulation [102], which divides faults into groups. Group g_0 contains undetected faults. Groups with more than one fault contain the faults that are mutually undistinguished (or undiagnosed). Thus, circuit s27 has 12 such groups and the largest of those groups has 7 faults (see columns 6 and 7). Similarly, s5378 has 400 multi-fault undiagnosed groups, the largest one containing 9 faults. In [38] instead of targeting all transition faults they randomly choose 1000 faults and extract those pairs that cannot be distinguished by a detection test set. These pairs then serve as the starting set for their diagnostic test generation flow. For s38584 the CPU time reported in [38] is 174649 seconds, whereas ours is 14841 seconds for the same circuit (Table 5.2). Note that CPU times reported here cannot be compared directly (hardware configuration is not reported in [38]).

Table 5.2 shows a comparison between our work and [38]. Note that the CPU time in our work is not optimized. For every fault pair the CUT is flattened and the data structure reconstructed. This huge overhead can be avoided by modifying the data structure incrementally. Thus the proposed algorithm has great potential in efficiency. Since only number of distinguished fault pairs are reported in [38], the coverage is calculated as number of distinguished fault pairs divided by number of total fault pairs (499,500) for their results. In [38], only 1000 faults are chosen randomly as the starting fault set in their experiment, whereas we are targeting all collapsed fault (56226). A total of 2188 test vectors (detection and diagnosis) are generated with our DATPG system targeting all transition faults, and 2703 test vectors are generated for 1000 random transition faults in [38].

The purpose of diagnostic test generation is to derive exclusive tests that will provide pair-wise diagnosis of faults within groups. This is done by modeling a pair of transition faults as two stuck-at faults, using the technique of Figure 5.4 and then using a single stuck-at fault representation for those two faults [102]. Additional tests obtained for fault pairs formed within each multi-fault group are listed in column 8 of Table 5.1. The corresponding diagnostic coverage (*DC*) is given in column 9. For example, 18 tests were generated for s27 raising *DC* to 97.8%. There was only one undiagnosed fault group left (column 10) and it contained two faults (column 11).

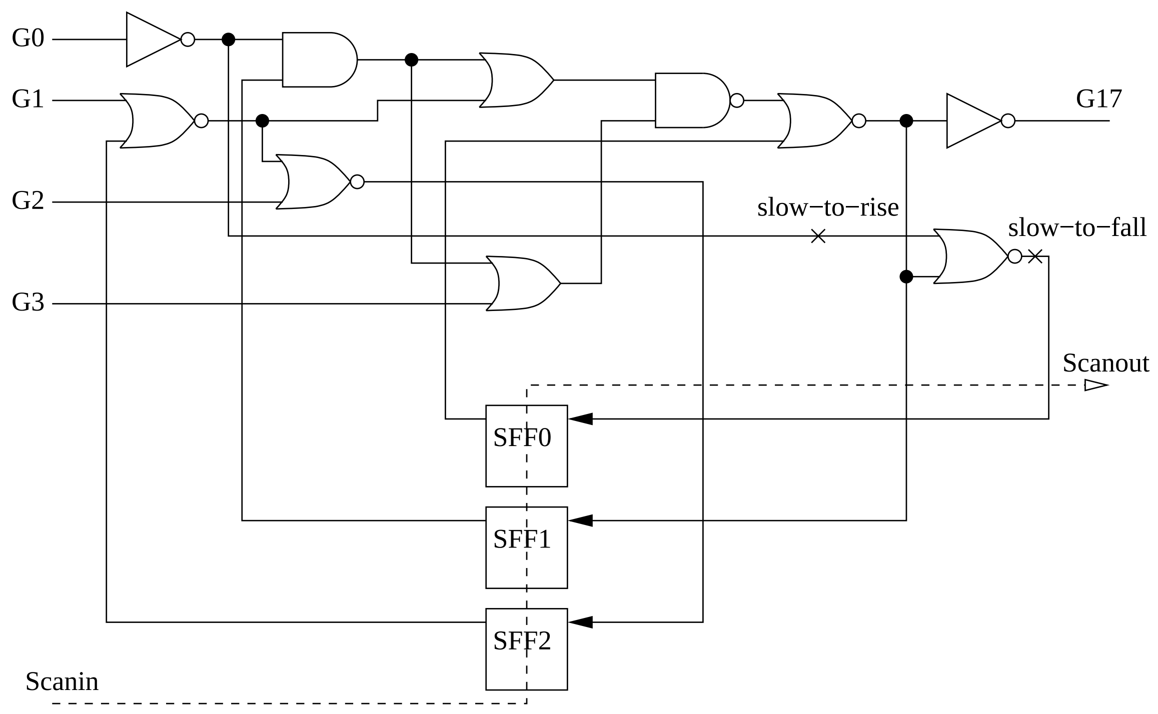


Figure 5.7: Circuit s27 showing the fault pair left undiagnosed after simulation of 11+18 LOC tests of Table 5.1. These faults are functionally equivalent though they are diagnosable in LOS mode.

The two undiagnosed faults of s27 are shown in Figure 5.7. Using a two time-frame combinational model, we determined that these two faults cannot be distinguished by any LOC test. Because the functional operation of the circuit is constrained to a subset of conditions allowed during the LOC testing, these two faults can be considered functionally equivalent. That will make $DC = 100\%$ in column 9. At the present stage of our investigation, such fault equivalence checking is not automated. Once we have enhanced such capability, we hope to analyze the undiagnosed fault groups in column 10 for all circuits. We have verified that with an added LOS test all faults in s27 can be diagnosed.

In [5] a method was proposed to model a path delay fault with a single stuck-at fault. The detection condition of the delay fault is equivalent to the detection condition of the modeling stuck-at fault. Equipped with this algorithms our DATPG can also be used to generate exclusive test for path delay faults.

Generally speaking, any fault if its faulty behavior can be mapped to a single stuck-at fault with additional modeling logic gate(s), a conventional stuck-at ATPG tool can be used for detection or diagnostic test generation for such faults. Delay effect can be modeled with flip-flops. Incorrect logic can be modeled with primitive gates (AND, OR, NOT). A multiplexer is added to choose between fault free behavior and faulty behavior (detection test) or between two different faulty behaviors (exclusive test). The two faulty behaviors do not need to come from same fault models. Thus mixed fault model diagnosis is easy to implement for the proposed DATPG system.

Chapter 6
Conclusion and Future Work

In this chapter we conclude our work and give some examples for future development of this DATPG system.

6.1 Conclusion

The proposed DATPG generates tests to distinguish fault pairs, i.e., two faults must have different output responses. Given a fault pair, by modifying circuit netlist a new single fault is modeled. Then we use a conventional ATPG to target that fault. If a test is generated it distinguishes the given fault pair. A fast diagnostic fault simulation algorithm is implemented to find undistinguished fault pairs from a fault list for a given test vector set. We define a new diagnostic coverage (DC) metric [102] as the ratio of the number of fault groups (faults within a group are not distinguished from each other but across groups are distinguished) to the number of total faults. The diagnostic ATPG system starts by first generating conventional fault coverage vectors. Those vectors are then simulated to determine the DC, followed by repeated applications of diagnostic test generation and simulation.

In Chapter 4 we construct a DATPG system for a combinational circuit with stuck-at faults [102]. This DATPG system augments the ability of conventional ATPG to generate exclusive tests without increasing the complexity. Thus, a specially designed DATPG tool [35] is no longer needed. Chapter 5 shows the potential to distinguish between arbitrary fault models, thus making it very practical for diagnosing actual physical defects.

The Boolean analysis in Chapter 4 provides a theoretical proof of the one-copy ATPG model for generating an exclusive test that also provides greater insight into the process than previous work [93]. An efficient novel diagnostic fault simulator [103] is integrated with this ATPG model to form a DATPG system that, to the best of our knowledge, has not been reported prior to this research. This simulation algorithm also makes it possible to construct rather compact fault dictionaries. Our objective, however, is to generate tests with high DC. If necessary, any previous dictionary compaction method can be used for further test compaction without affecting DC [25, 39, 50, 68, 74, 80]. The DATPG system can be modified to target multiple and/or mixed fault models. We provide details of new methods to generate exclusive tests for transition faults [104].

The stuck-at fault models of transition faults derived in this work are significantly more compact than those previously published [38]. Combined with diagnostic fault simulation and test generation algorithms similar to those for stuck-at faults [102], the new transition fault models provide potentially very effective ATPG methodology. Delay fault diagnosis is important in the characterization of modern VLSI devices and a high diagnostic coverage of tests is desirable. Whether or not the tests have an adequate diagnostic coverage cannot be ascertained unless we have an effective tool for identifying fault equivalence [13, 37, 71, 75, 76, 77]. The present work provides the possibility of doing so entirely by combinational ATPG programs. This is because we have shown that once an exclusive test for any type of faults is modeled as a single stuck-at fault, the problem of fault equivalence is solved by redundancy identification. The problem, however, has high complexity. As a result heuristic methods are found to be useful [7, 34, 42, 60]. Our ongoing research is exploring this aspect. That will give transition fault testing the same level of maturity as enjoyed by stuck-at faults. Another direction for the future is to make the diagnostic tests specific to small delay defects (SDD), i.e., derive tests to detect transition faults, each through the longest sensitizable path through its site [56].

6.2 Future Work

6.2.1 Exclusive Tests for Other Fault Models

Tests generated by the system of Chapter 4 have high diagnostic capability but they are specific to classical stuck-at fault model. Because this fault model is implementation independent, such tests serve as a good starting point. As a typical VLSI design progresses, technology specific data on non-classical fault models becomes available. Besides, the characterization phase may identify faults that are not diagnosable by the pre-generated tests. In such cases, it becomes necessary to generate more tests. This section illustrates the possibility of using the core ATPG algorithm of Section 4.2 for a non-classical bridging fault. Diagnostic tests of arbitrary fault have been attempted using Boolean satisfiability (SAT) [16] and time-frame expansion methods [38]. The SAT procedure uses a tuple representation for a variety of faults that includes time-frame information [16]. The SAT solution requires large CPU time and in some cases (e.g., c6288) high space complexity. Because complexity of these solutions is high there is scope for efficient methods.

The algorithm of Section 4.2, although derived for stuck-at faults, is adaptable to the so-called non-classical faults (bridging, transition, etc.) [4, 20, 43]. The trivially small example of Figure 6.1 illustrates the point. This textbook circuit [20] has a collapsed set of six stuck-at faults diagnosed by four vectors, (a, b, c): 010, 011, 100, 110. Table 6.1 gives a fault dictionary that associates a binary vector called a syndrome (signature) with each fault. A row in this table corresponds to a fault and a "1" entry under a test means that the corresponding fault is detectable by the test. Ignore the last column and last row of Table 6.1 for now. The four-bit vector 0001 in the row of a s-a-0 fault is the syndrome for that fault. Since every fault has a unique four-bit syndrome, all stuck-at faults can be completely diagnosed by four vectors. Consider a non-classical fault, AND bridge (a, c). That means in the faulty circuit signals a and c are replaced by $a \wedge c$. An analysis of the fault-free circuit shown on the upper side of Figure 6.1 gives a syndrome 0101, which matches that of e s-a-0. Although the

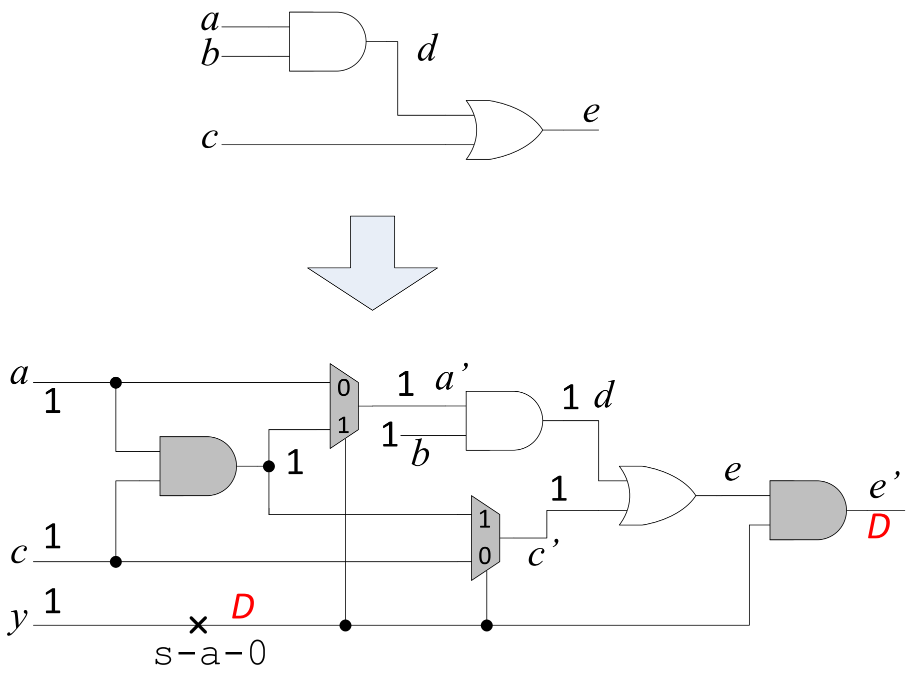


Figure 6.1: Exclusive test for e s-a-0 and AND bridge (a, c) faults.

bridging fault is distinguishable from all other stuck-at faults, it cannot be isolated from e s-a-0 by the four vectors. We therefore need an exclusive test for faults AND bridge (a, c) and e s-a-0.

The circuit on the bottom of Figure 6.1 shows the construction of Section 4.2. Signals a, c, e that are affected by faults have been replaced by a', c', e'. When $y = 0$, $a' = a$ and $c' = c$ have their correct values, but $e' = 0$ emulates the e s-a-0 fault. When $y = 1$, $e' = e$, but $a' = c' = a \wedge c$. We derive a test for y s-a-0, which gives us an exclusive test $(a, b, c) = 111$. This detects e s-a-0, but does not detect AND bridge (a, c). Adding this test provides us the last column in Table 6.1.

The number of bridging faults, though large, can be reduced depending upon how much physical data is available [32, 85, 86]. Recent results by our colleagues at Auburn University show a drastic reduction when capacitive coupling is used [89]. The coverage of bridging faults by a set of vectors usually remains close to that of stuck-at faults. However, diagnostic testing requires distinguishing a bridging fault from single or multiple stuck-at [48], other bridging, transition and other faults. Our future research will target such tests.

Table 6.1: Fault dictionary for the AND-OR circuit shown on the upper side in Figure 6.1.

Collapsed Faults	Test syndrome				
	010	011	100	110	111
a sa0	0	0	0	1	0
a sa1	1	0	0	0	0
b sa1	0	0	1	0	0
c sa0	0	1	0	0	0
c sa1	1	0	1	0	0
e sa0	0	1	0	1	1
AND bridge (a,c)	0	1	0	1	0

6.2.2 Diagnostic Test Pattern Compaction

Compactness of tests is desirable because it translates into efficiency of test and diagnosis. One recent research [80, 81, 82, 83] explores two principal ideas. First, mathematical optimization techniques like integer linear program (ILP) provide absolute minimal results. High complexity can be reduced by prudent choices of heuristics [30, 40, 44, 45, 46, 72]. Second, fault independence has been used for finding compact detection tests [8, 10, 28, 29, 45]. Two faults are called independent if they have no common test. The largest group of mutually independent faults is a lower bound on the tests required for 100% detection. The tests generated for independent faults may also have superior diagnostic properties and we will examine those.

The proposed ILP based compaction approach has high computational complexity and long CPU time. With our DATPG system it either eliminates the need for compaction with N-detect test or greatly reduces the CPU time for ILP run, because a N-detect test set is usually much larger compared to our generated test set. Reverse/random order simulation of generated vector set can also provide some compaction. If no new faults are detected *and* no new fault groups are formed, the vector in reverse/random simulation can be dropped.

6.2.3 Diagnosis of Real Defects

The ultimate goal of diagnosis is to find the location and nature of a real defect causing a functional (electrical) failure. Modeled faults, on the other hand, are analyzable entities, but not all real defects actually map onto modeled faults. The idea behind using multiple fault models, suggested in Section 6.2.1, is to obtain tests that will activate and detect the presence of most real defects. Diagnosis, being an analytical procedure, must use fault models and, therefore, exact identification of many defects may not be possible. A possible direction would be to use electrical tests to narrow down the possible location of the defect. The purpose is to facilitate additional procedures, such as optical or electron microscopes, to identify the actual cause that may be a mechanical or quantum mechanical defect. Recent work has used stuck-at and transition faults to diagnose faulty nets, which may be then closely examined for real defects [105, 106].

Bibliography

[1] "22 nanometer." `http://en.wikipedia.org/wiki/22_nanometer`, 2012. [Online; accessed 03-Mar-2012].

[2] "Race condition." `http://en.wikipedia.org/wiki/Race_condition`, 2012. [Online; accessed 17-Mar-2012].

[3] M. Abramovici and M. A. Breuer, "Multiple Fault Diagnosis in Combinational Circuits Based on an Effect-Cause Analysis," *IEEE Transactions on Computing*, vol. C-29, no. 6, pp. 451–460, June 1980.

[4] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design.* Piscataway, New Jersey: IEEE Press, 1994.

[5] P. Agrawal, V. D. Agrawal, and S. C. Seth, "Generating Tests for Delay Faults in Nonscan Circuits," *IEEE Design & Test of Computers*, vol. 10, no. 1, pp. 20–28, Mar. 1993.

[6] V. D. Agrawal, D. H. Baik, Y. C. Kim, and K. K. Saluja, "Exclusive Test and its Applications to Fault Diagnosis," in *Proc. 16th International Conf. VLSI Design*, Jan. 2003, pp. 143–148.

[7] V. D. Agrawal, M. L. Bushnell, and Q. Lin, "Redundancy Identification using Transitive Closure," in *Proc. Fifth IEEE Asian Test Symp.*, Nov. 1996, pp. 4–9.

[8] V. D. Agrawal and A. S. Doshi, "Concurrent Test Generation," in *Proc. 14th IEEE Asian Test Symp.*, Dec. 2005, pp. 294–297.

[9] V. D. Agrawal, A. V. S. S. Prasad, and M. V. Atre, "Fault Collapsing via Functional Dominance," in *Proc. International Test Conf.*, 2003, pp. 274–280.

[10] S. B. Akers, C. Joseph, and B. Krishnamurthy, "On the Role of Independent Fault Sets in the Generation of Minimal Test Sets," in *Proc. International Test Conf.*, 1987, pp. 1100–1107.

[11] H. Al-Asaad and J. P. Hayes, "Logic Design Validation via Simulation and Automatic Test Pattern Generation," *J. Electronic Testing: Theory and Applications*, vol. 16, no. 6, pp. 575–589, Dec. 2000.

[12] M. E. Amyeen, W. K. Fuchs, I. Pomeranz, and V. Boppana, "Implication and Evaluation Techniques for Proving Fault Equivalence," in *Proc. 17th IEEE VLSI Test Symp.*, 1999, pp. 201–207.

[13] M. E. Amyeen, W. K. Fuchs, I. Pomeranz, and V. Boppana, "Fault Equivalence Identification in Combinational Circuits Using Implication and Evaluation Techniques," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 7, pp. 922–936, July 2003.

[14] K. Banerjee, D.-Y. Kim, A. Amerasekera, C. Hu, S. S. Wong, and K. E. Goodson, "Microanalysis of VLSI Interconnect Failure Modes under Short-Pulse Stress Conditions," in *International Reliability Physics Symposium*, 2000, pp. 283–288.

[15] B. Benware, C. Schuermyer, S. Ranganathan, R. Madge, N. Tamarpalli, K.-H. Tsai, and J. Rajski, "Impact of Multiple-Detect Test Patterns on Product Quality," in *Proc. International Test Conf.*, 2003, pp. 1031–1040.

[16] N. K. Bhatti and R. D. Blanton, "Diagnostic Test Generation for Arbitrary Faults," in *Proc. International Test Conf.*, 2006. Paper 19.2.

[17] D. C. Bossen and S. J. Hong, "Cause-Effect Analysis for Multiple Fault Detection in Combinational Networks," *IEEE Trans. Computers*, vol. C-20, no. 11, pp. 1252–1257, Nov. 1971.

[18] F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," in *Proc. IEEE International Symp. Circuits and Systems*, May 1989, pp. 1929–1934, vol. 3.

[19] F. Brglez, P. Pownall, and R. Hum, "Accelerated ATPG and Fault Grading via Testability Analysis," in *Proc. IEEE International Symp. Circuits and Systems*, June 1985, pp. 695–698.

[20] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits*. Boston: Springer, 2000.

[21] P. Camurati, A. Lioy, P. Prinetto, and M. S. Reorda, "A Diagnostic Test Pattern Generation Algorithm," in *Proc. International Test Conf.*, 1990, pp. 52–58.

[22] H. Y. Chang, E. Manning, and G. Metze, *Fault Diagnosis of Digital Systems*. New York: Wiley-Interscience, 1970.

[23] S.-C. Chen and J. M. Jou, "Diagnostic Fault Simulation for Synchronous Sequential Circuits," *IEEE Trans. Computer-Aided Design*, vol. 16, no. 3, pp. 299–308, Mar. 1997.

[24] W. T. Cheng and T. J. Chakraborty, "GENTEST: An Automatic Test Generation System for Sequential Circuits," *Computer*, vol. 22, no. 4, pp. 43–49, Apr. 1989.

[25] B. Chess and T. Larrabee, "Creating Small Fault Dictionaries," *IEEE Trans. Computer-Aided Design*, vol. 18, no. 3, pp. 346–356, Mar. 1999.

[26] A. L. Crouch, *Design for Test for Digital ICs and Embedded Core Systems*. New Jersey: Prentice Hall, 1999.

[27] B. Dervisoglu and G. Stong, "Design for Testability: Using Scanpath Techniques for Path-Delay Test and Measurement," in *Proc. International Test Conf.*, 1991, pp. 365–347.

[28] A. S. Doshi, "Independence Fault Collapsing and Concurrent Test Generation," Master's thesis, Auburn University, ECE Department, May 2006.

[29] A. S. Doshi and V. D. Agrawal, "Independence Fault Collapsing," in *Proc. 9th VLSI Design & Test Symp. (VDAT'05)*, Aug. 2005, pp. 357–366.

[30] P. Drineas and Y. Makris, "Test Sequence Compaction Through Integer Programming," in *Proc. International Conf. Computer Design*, 2003, pp. 380–386.

[31] R. D. Eldred, "Test Routines Based on Symbolic Logical Statements," *Journal of the ACM*, vol. 6, no. 1, pp. 33–36, Jan. 1959.

[32] F. J. Ferguson and T. Larrabee, "Test Pattern Generation for Realistic Bridge Faults in CMOS ICs," in *Proc. IEEE International Test Conf.*, 1991, pp. 492–499.

[33] J. M. Galey, R. E. Norby, and J. P. Roth, "Techniques for the Diagnosis of Switching Circuit Failures," in *Proc. Second Annual Symp. on Switching Circuit Theory and Logical Design*, 1961, pp. 152–160.

[34] V. Gaur, V. D. Agrawal, and M. L. Bushnell, “A New Transitive Closure Algorithm with Application to Redundancy Identification,” in *Proc. 1st International Workshop on Electronic, Design and Test Applications (DELTA'02)*, Jan. 2002, pp. 496–500.

[35] T. Gruning, U. Mahlstedt, and H. Koopmeiners, “DIATEST: A Fast Diagnostic Test Pattern Generator for Combinational Circuits,” in *Proc. IEEE International Conf. Computer-Aided Design*, 1991, pp. 194–197.

[36] M. C. Hansen, H. Yalcin, and J. P. Hayes, “Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering,” *IEEE Design & Test of Computers*, vol. 16, no. 3, pp. 72–80, 1999.

[37] I. Hartanto, V. Boppana, and W. K. Fuchs, “Diagnostic Fault Equivalence Identification Using Redundancy Information & Structural Analysis,” in *Proc. International Test Conf.*, 1996, pp. 20–25.

[38] Y. Higami, Y. Kurose, S. Ohno, H. Yamaoka, H. Takahashi, Y. Takamatsu, Y. Shimizu, and T. Aikyo, “Diagnostic Test Generation for Transition Faults Using a Stuck-at ATPG Tool,” in *Proc. International Test Conf.*, 2009. Paper 16.3.

[39] Y. Higami, K. K. Saluja, H. Takahashi, S. Kobayashi, and Y. Takamatsu, “Compaction of Pass/Fail-based Diagnostic Test Vectors for Combinational and Sequential Circuits,” in *Proc. Asia and South Pacific Design Automation Conf. (ASPDAC)*, 2006, pp. 75–80.

[40] D. S. Hochbaum, “An Optimal Test Compression Procedure for Combinational Circuits,” *IEEE Trans. Computer-Aided Design*, vol. 15, no. 10, pp. 1294–1299, Oct. 1996.

[41] F. Hsu, K. Butler, and J. H. Patel, “A Case Study on the Implementation of Illinois Scan Architecture,” in *Proc. International Test Conf.*, 2001, pp. 538–547.

[42] M. A. Iyer and M. Abramovici, “Low-Cost Redundancy Identification for Combinational Circuits,” in *Proc. 7th International Conf. VLSI Design*, Jan. 1994, pp. 268–273.

[43] N. K. Jha and S. K. Gupta, *Testing of Digital Systems*. London, United Kingdom: Cambridge University Press, 2002.

[44] K. R. Kantipudi, “Minimizing N-Detect Tests for Combinational Circuits,” Master's thesis, Auburn University, ECE Department, May 2007.

[45] K. R. Kantipudi and V. D. Agrawal, “On the Size and Generation of Minimal N-Detection Tests,” in *Proc. 19th International Conf. VLSI Design*, Jan. 2006, pp. 425–430.

[46] K. R. Kantipudi and V. D. Agrawal, “A Reduced Complexity Algorithm for Minimizing N-Detect Tests,” in *Proc. 20th International Conf. VLSI Design*, Jan. 2007, pp. 492–497.

[47] X. Kavousianos and K. Chakrabarty, “Generation of Compact Stuck-At Test Sets Targeting Unmodeled Defects,” *IEEE Trans. Computer-Aided Design*, vol. 30, no. 5, pp. 787–791, May 2011.

[48] Y. C. Kim, V. D. Agrawal, and K. K. Saluja, “Multiple Faults: Modeling, Simulation and Test,” in *Proc. 15th International Conf. VLSI Design*, Jan. 2002, pp. 592–597.

[49] D. Lavo, B. Chess, T. Larrabee, and F. J. Ferguson, “Diagnosing Realistic Bridging Faults with Single Stuck-at Information,” *IEEE Trans. Computer-Aided Design*, vol. 17, no. 3, pp. 255–268, Mar. 1998.

[50] D. Lavo and T. Larrabee, "Making Cause-Effect Cost Effective: Low-Resolution Fault Dictionaries," in *Proc. International Test Conf.*, 2001, pp. 278–286.

[51] D. B. Lavo, *Comprehensive Fault Diagnosis of Combinational Circuits.* PhD thesis, University of California Santa Cruz, Sept. 2002.

[52] H. K. Lee and D. S. Ha, *On the Generation of Test Patterns for Combinational Circuits.* Tech. Report 12-93, Dept. of Elec. Eng., Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1993.

[53] H. K. Lee and D. S. Ha, "HOPE: An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits," *IEEE Trans. Computer-Aided Design*, vol. 15, no. 9, pp. 1048–1058, Sept. 1996.

[54] Y.-C. Lin and K.-T. Cheng, "Multiple-Fault Diagnosis based on Single-Fault Activation and Single-Output Observation," in *Proc. Design, Automation and Test in Europe*, 2006, pp. 424–429.

[55] Y.-C. Lin, F. Lu, and K.-T. Cheng, "Multiple-Fault Diagnosis based on Adaptive Diagnostic Test Pattern Generation," *IEEE Trans. on CAD*, vol. 26, no. 5, pp. 932–942, May 2007.

[56] A. K. Majhi, V. D. Agrawal, J. Jacob, and L. M. Patnaik, "Line Coverage of Path Delay Faults," *IEEE Trans. VLSI Systems*, vol. 8, pp. 610–614, Oct. 2000.

[57] W. Maly, "Realistic Fault Modeling for VLSI Testing," in *Proc. 24th Design Automation Conf.*, 1987, pp. 173–180.

[58] E. J. Marinissen, V. Iyengar, and K. Chakrabarty, "A Set of Benchmarks for Modular Testing of SOCs," in *Proc. International Test Conf.*, 2002. Paper 19.1.

[59] E. J. McCluskey and C.-W. Tseng, "Stuck-Fault Tests vs. Actual Defects," in *Proc. International Test Conf.*, 2000, pp. 336–342.

[60] V. J. Mehta, "Redundancy Identification in Logic Circuits using Extended Implication Graph and Stem Unobservability Theorems ," Master's thesis, Rutgers University, ECE Department, May 2003.

[61] Mentor Graphics, *FastScan and FlexTest Reference Manual*, 2004.

[62] Mentor Graphics, *DFTAdvisor Reference Manual*, 2009.

[63] Mentor Graphics, *Scan and ATPG Process Guide*, 2009.

[64] F. Nekoogar, *Timing Verification of Application-Specific Integrated Circuits (ASICs).* Upper Saddle River, New Jersey: Prentice Hall, 1999.

[65] V. P. Nelson, H. T. Nagle, B. D. Carroll, and J. D. Irwin, *Digital Logic Circuit Analysis & Design.* Englewood Cliffs, New Jersey: Prentice Hall, 1995.

[66] J. H. Patel, "Illinois Scan Architecture." `http://courses.engr.illinois.edu/ece543/docs/IllinoisScan_6_per_page.pdf`, 2005. [Online; accessed 12-Feb-2012].

[67] W. A. Pleskacz, V. Stopjakova, T. Borejko, A. Jutman, and A. Walkanis, "DefSim: A Remote Laboratory for Studying Physical Defects in CMOS Digital Circuits," *IEEE Trans. on Industrial Electronics*, vol. 55, no. 6, pp. 2405–2415, June 2008.

[68] I. Pomeranz and S. M. Reddy, "On the Generation of Small Dictionaries for Fault Location," in *Proc. IEEE International Conf. Computer-Aided Design*, 1992, pp. 272–278.

[69] I. Pomeranz and S. M. Reddy, "On Correction of Multiple Design Errors," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 2, pp. 255–264, Feb. 1995.

[70] I. Pomeranz and S. M. Reddy, "Location of Stuck-at Faults and Bridging Faults Based on Circuit Partitioning," *IEEE Trans. on Computers*, vol. 47, pp. 1124–1135, Oct. 1998.

[71] I. Pomeranz and S. M. Reddy, "Fault Collapsing for Transition Faults Using Extended Transition Faults," in *Proc. 11th IEEE European Test Conf.*, 2006, pp. 760–769.

[72] P. Raghavan and C. D. Thompson, "Randomized Rounding: A Technique for Provably Good Algorithms and Algorithmic Proofs," *Combinatorica*, vol. 7, no. 4, pp. 365–374, Oct. 1987.

[73] E. M. Rudnick, W. K. Fuchs, and J. H. Patel, "Diagnostic Fault Simulation of Sequential Circuits," in *Proc. International Test Conf.*, 1992, pp. 178–186.

[74] P. G. Ryan, W. K. Fuchs, and I. Pomeranz, "Fault Dictionary Compression and Equivalence Class Computation for Sequential Circuits," in *Proc. Int. Conf. on Computer-Aided Design*, 1993, pp. 508–511.

[75] R. K. K. R. Sandireddy, "Hierarchical Fault Collapsing for Logic Circuits," Master's thesis, Auburn University, ECE Department, May 2005.

[76] R. K. K. R. Sandireddy and V. D. Agrawal, "Diagnostic and Detection Fault Collapsing for Multiple Output Circuits," in *Proc. Design, Automation and Test in Europe (DATE'05)*, Mar. 2005, pp. 1014–1019.

[77] R. K. K. R. Sandireddy and V. D. Agrawal, "Using Hierarchy in Design Automation: The Fault Collapsing Problem," in *Proc. 11th VLSI Design & Test Symp. (VDAT'07)*, Aug. 2007, pp. 174–184.

[78] J. Savir and S. Patil, "Scan-Based Transition Test," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 8, Aug. 1993.

[79] J. Savir and S. Patil, "On Broad-Side Delay Test," in *VLSI Test Symp.*, 1994, pp. 284–290.

[80] M. A. Shukoor, "Fault Detection and Diagnostic Test Set Minimization," Master's thesis, Auburn University, ECE Department, May 2009.

[81] M. A. Shukoor and V. D. Agrawal, "A Primal-Dual Solution to the Minimal Test Generation Problem," in *Proc. 12th VLSI Design and Test Symp.*, 2008, pp. 169–179.

[82] M. A. Shukoor and V. D. Agrawal, "A Two Phase Approach for Minimal Diagnostic Test Set Generation," in *Proc. 14th IEEE European Test Symp.*, May 2009, pp. 115–120.

[83] M. A. Shukoor and V. D. Agrawal, "Diagnostic Test Set Minimization and Full-Response Fault Dictionary," *Jour. Electronic Testing: Theory and Applications*, vol. 28, no. 2, Apr. 2012.

[84] N. Sridhar and M. S. Hsiao, "On Efficient Error Diagnosis of Digital Circuits," in *Proc. International Test Conf.*, 2001, pp. 678–687.

[85] C. Stroud, J. Bailey, J. Emmert, D. Nickolic, and K. Chhor, "Bridging Fault Extraction from Physical Design Data for Manufacturing Test Development," in *Proc. IEEE International Test Conf.*, 2000, pp. 760–769.

[86] C. Stroud, J. Emmert, and J. Bailey, "A New Bridging Fault Model for More Accurate Fault Behavior," in *Proc. IEEE Automatic Test Conf.*, 2000, pp. 481–485.

[87] C. E. Stroud, *A Designer's Guide to Built-In Self-Test.* New York: Kluwer Academic, 2002.

[88] Synopsys, *TetraMAX ATPG User Guide*, 2000.

[89] J. M. Tacey, M. A. Lusco, J. Qin, N. S. DaCunha, and F. F. Dai, "Weighted Bridging Fault Coverage Using Capacitance Extraction," in *43rd Southeastern Symp. on System Theory*, 2011, pp. 249–254.

[90] H. Takahashi, K. O. Boateng, K. K. Saluja, and Y. Takamatsu, "On Diagnosing Multiple Stuck-at Faults Using Multiple and Single Fault Simulation in Combinational Circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 3, pp. 362–368, Mar. 2002.

[91] Y. Takamatsu, H. Takahashi, Y. Higami, T. Aikyo, and K. Yamazaki, "Fault Diagnosis on Multiple Fault Models by Using Pass/Fail Information," *IEICE Transactions on Information and Systems*, vol. E91-D, no. 3, pp. 675–682, 2008.

[92] G. van Rossum and F. L. Drake, Jr., editors, *Python Tutorial Release 2.6.3.* docs@python.org: Python Software Foundation, Oct. 2009.

[93] A. Veneris, R. Chang, M. S. Abadir, and M. Amiri, "Fault Equivalence and Diagnostic Test Generation Using ATPG," in *Proc. Int. Symp. Circuits and Systems*, 2004, pp. 221–224.

[94] S. Venkataraman and S. B. Drummonds, "Poirot: Applications of a Logic Fault Diagnosis Tool," *IEEE Design & Test of Computers*, vol. 3, pp. 19–30, Jan. 2001.

[95] S. Venkataraman, I. Hartanto, and W. K. Fuchs, "Rapid Diagnostic Fault Simulation of Stuck-at Faults in Sequential Circuits using Compact Lists," in *Proc. 32nd ACM/IEEE Design Automation Conf.*, 1995, pp. 133–138.

[96] J. A. Waicukauski and E. Lindbloom, "Failure Diagnosis of Structured VLSI," *IEEE Design and Test of Computers*, vol. 6, no. 4, pp. 49–60, Aug. 1989.

[97] L.-T. Wang, C.-W. Wu, and X. Wen, *VLSI Test Principles and Architectures: Design for Testability.* San Francisco, CA: Morgan Kaufmann, 2006.

[98] N. Yogi, *Spectral Methods for Testing of Digital Circuits.* PhD thesis, Auburn University, ECE Department, Aug. 2009.

[99] N. Yogi and V. D. Agrawal, "N-Model Tests for VLSI Circuits," in *Proc. 40th Southeastern Symp. System Theory*, 2008, pp. 242–246.

[100] X. Yu, M. E. Amyeen, S. Venkataraman, R. Guo, and I. Pomeranz, "Concurrent Execution of Diagnostic Fault Simulation and Equivalence Identification During Test Generation," in *Proc. 21st IEEE VLSI Test Symp.*, 2003, pp. 351–356.

[101] X. Yu, J. Wu, and E. M. Rudnick, "Diagnostic Test Generation for Sequential Circuits," in *Proc. International Test Conf.*, 2000, pp. 226–234.

[102] Y. Zhang and V. D. Agrawal, "A Diagnostic Test Generation System," in *Proc. International Test Conf.*, 2010. Paper 12.3.

[103] Y. Zhang and V. D. Agrawal, "An Algorithm for Diagnostic Fault Simulation," in *Proc. 11th IEEE Latin-American Workshop*, 2010.

[104] Y. Zhang and V. D. Agrawal, "Reduced Complexity Test Generation Algorithms for Transition Fault Diagnosis," in *International Conference on Computer Design (ICCD)*, Oct. 2011, pp. 96–101.

[105] L. Zhao, "Net Diagnosis Using Stuck-at and Transition Fault Models," Master's thesis, Auburn University, ECE Department, Dec. 2011.

[106] L. Zhao and V. D. Agrawal, "Net Diagnosis Using Stuck-at and Transition Fault Models," in *Proc. 30th IEEE VLSI Test Symp.*, Apr. 2012.