University of California
Santa Cruz

# Diagnostic Test Pattern Generation and the Creation of Small Fault Dictionaries

A thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Computer Engineering

by

Brian Chess

June 1995

The thesis of Brian Chess is approved:

F. Joel Ferguson

John M. Acken

Darrell D. E. Long

Dean of Graduate Studies and Research

Copyright © by

Brian Chess

1995

# Contents

**Abstract** **viii**

**Acknowledgments** **ix**

**1. Introduction** **1**

**2. Diagnostic Test Pattern Generation** **4**

2.1 DTPG for stuck-at faults . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

2.2 Stuck-at DTPG results . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

2.3 DTPG for $I_{DDQ}$ detectable bridging faults . . . . . . . . . . . . . . . . . . . . 13

2.4 Bridge $I_{DDQ}$ DTPG results . . . . . . . . . . . . . . . . . . . . . . . . . . 14

**3. Dictionary organization** **19**

3.1 Popular dictionary organizations . . . . . . . . . . . . . . . . . . . . . . . 19

3.2 The Error Set . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22

3.3 Dictionary organization using error sets . . . . . . . . . . . . . . . . . . . . 24

3.4 Further dictionary compaction using error sets . . . . . . . . . . . . . . . . 25

3.5 Circuit and test set characterization . . . . . . . . . . . . . . . . . . . . . 27

3.6 Dictionary results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28

**4. Diagnosis of faults not modeled by the dictionary** **32**

**5. Conclusions** **36**

**References** **37**

# List of Figures

2.1 A small example circuit. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

2.2 Testing for E stuck-at 1 and distinguishing between E and F stuck-at 1. . . 7

2.3 The 8 stuck-at 0 faults on the fanout stems are indistinguishable . . . . . . 12

2.4 A differentiating $I_{DDQ}$ test detects X or Y but not both. . . . . . . . . . . . 14

2.5 A bridging fault that will cause excess $I_{DDQ}$ current for 97% of randomly applied tests. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

2.6 A bridging fault that will always cause excess $I_{DDQ}$ current. . . . . . . . . . 18

3.1 The C17. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19

3.2 The diagnostic tree . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22

3.3 The number of data items in an error set dictionary tracks the number of fault detections the dictionary stores. . . . . . . . . . . . . . . . . . . . . . . 31

4.1 Diagnostic expectation for four dictionaries. (The diagnostic expectat ion for the drop-when-distinguished dictionary is identical to the diagnostic ex pectation for the full dictionary.) . . . . . . . . . . . . . . . . . . . . . . . 33

4.2 Diagnostic failure for four dictionaries. . . . . . . . . . . . . . . . . . . . . . 35

# List of Tables

2.1 Diagnostic Test Pattern Generation for stuck-at faults. . . . . . . . . . . . . 10

2.2 Number of stuck-at fault equivalence classes by size before DTPG. . . . . . 11

2.3 Number of stuck-at fault equivalence classes by size after DTPG. . . . . . . 11

2.4 Diagnostic expectation before and after stuck-at DTPG . . . . . . . . . . . 12

2.5 DTPG for $I_{DDQ}$ detectable bridging faults. . . . . . . . . . . . . . . . . . . 15

2.6 Diagnostic expectation for the $I_{DDQ}$ test sets. . . . . . . . . . . . . . . . . . 15

2.7 Test set size. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

2.8 Equivalence classes for $I_{DDQ}$ detectable bridging faults by size before DTPG. 16

2.9 Equivalence classes for $I_{DDQ}$ detectable bridging faults by size after DTPG. 17

2.10 Equivalence classes for $I_{DDQ}$ detectable bridging faults by size with a complete $I_{DDQ}$ test set augmented with random patterns. . . . . . . . . . . . . . 17

2.11 Number of faults that are detected by every $I_{DDQ}$ test. . . . . . . . . . . . . 17

3.1 Partial fault list . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20

3.2 Test set . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20

3.3 Full response dictionary . . . . . . . . . . . . . . . . . . . . . . . . . . . 20

3.4 Pass/Fail dictionary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20

3.5 Compact dictionary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21

3.6 Detection dictionary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21

3.7 Vector dictionary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21

3.8 Error sets and final dictionary organization . . . . . . . . . . . . . . . . . . 25

3.9 Further compaction of the dictionary organization is achieved by dropping fully distinguished faults . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27

3.10 Circuit and test set characterization: the number of faults, tests, and primary outputs for each circuit followed by the size of the full dictionary in megabytes and the diagnostic expectation of the test set. . . . . . . . . . . . . . . . . . . 28

3.11 Lossless dictionary generation results . . . . . . . . . . . . . . . . . . . . . . 29

3.12 Drop-when-distinguished dictionary generation results . . . . . . . . . . . . 29

3.13 Dictionary size as a percentage of the size of the corresponding full dictionary for combinational and sequential circuits is given for Compact, the $DC2$ diagnostic tree, error set-based, and drop-when-distinguished error set-based dictionaries. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30

**Diagnostic Test Pattern Generation and the Creation of Small Fault Dictionaries**

*Brian Chess*

ABSTRACT

Fault diagnosis is an important part of failure analysis. The fault diagnosis procedure considered here involves selecting a set of faults and a set of tests, storing fault simulation results in a fault dictionary, and comparing dictionary entries against observed faulty behaviors. I investigate three aspects of this procedure. Just as automatic test pattern generation increases the fault coverage of a test set, diagnostic test pattern generation increases the diagnostic ability of a test set. I present a diagnostic test pattern generator for stuck-at faults and $I_{DDQ}$ detectable bridging faults in CMOS ICs. Fault simulation can generate enormous amounts of data. I present a new method of lossless dictionary compression and show how it can be combined with existing lossy compaction methods. If a fault dictionary has undergone lossy compaction, its ability to diagnose unmodeled faults may be severely curtailed. I demonstrate that dictionary quality metrics based solely on modeled faults do not reveal this flaw.

## Acknowledgments

For technical discussions, data, and analysis, I wish to thank Joel Ferguson, Nathaniel Hornblower, Tracy Larrabee, David Lavo, and Carl Roth. This work was supported by the Semiconductor Research Corporation and the National Science Foundation.

# 1. Introduction

Integrated circuit manufacturers are constantly trying to decrease the number of faulty parts they produce. By analyzing parts that fail production tests and determining each part's cause of failure, a manufacturer may be able to improve the circuit design or the manufacturing process. Determining the cause of failure for a circuit is known as *failure analysis*.

Failure analysis begins with *fault diagnosis*. The objective of fault diagnosis is to form a hypothesis that predicts the physical location of the failure on the chip. A failure analysis engineer uses fault diagnosis results to guide the search for the defect. If the diagnosis is imprecise, the engineer may have an overwhelming physical area to examine. Worse yet, if the diagnosis is incorrect and the engineer removes the upper layers of the faulty circuit, the actual defect may be destroyed and there will be no way to determine the cause of failure; it is important that a diagnosis be both precise and accurate.

Fault diagnosis begins with the selection of a *fault model*, which describes the behavior of the faults that will be used to explain a faulty circuit's behavior. The popular *single stuck-at* fault model assumes that the circuit contains a wire that has lost its ability to change values. The wire is *stuck-at 1* or *stuck-at 0*. The single stuck-at fault model is widely used because of its simplicity [23].

The majority of spot defects in modern CMOS technologies cause changes in the circuit description that result in shorts [11], which implies that many defects create bridging faults. The *bridging* fault model assumes that the outputs of two gates have been accidentally joined or bridged. Modeling the logic behavior of bridging faults is an option, but this

approach is complicated by the influence of relative drive strengths, bridge resistance, variable logic thresholds, and potential feedback loops [10]. The importance of these factors is compounded by the fact that accurate modeling of a fault's behavior is much more important for diagnosis than it is for production testing.

Another option for testing bridging faults is to monitor the quiescent power consumption of the chip under test. If a fault creates a conducting path from power to ground, the chip will consume an excess amount of current. The process of monitoring a chip's current during its quiescent state is known as $I_{DDQ}$ testing. Creating $I_{DDQ}$ tests for bridging faults is much easier than creating logic tests [12], and $I_{DDQ}$ testing can sometimes detect bridges that cannot be detected with logic tests. Diagnosing bridging faults with current measurements is an appealing option for manufacturers who already use $I_{DDQ}$ testing, but not all chips are designed to be $I_{DDQ}$ testable. Logic and $I_{DDQ}$ testing are both important fault diagnosis techniques.

Regardless of the fault models and detection methods used, the diagnosis procedure can take place as follows: First, all faults are simulated using the desired test set. Each simulated *fault signature* (the response of the circuit in the the presence of the fault) is stored in a *fault dictionary*. Next, the response of the actual faulty part is compared to each entry in the fault dictionary. If the faulty response is identical to a dictionary entry, the circuit is diagnosed as having the corresponding fault.

If a set of faults share the same signature, it is impossible to distinguish between them. These faults are said to form an *equivalence class* under the applied test set. The best diagnostic test set will differentiate between all faults that are distinguishable.

It is possible to create a good diagnostic test set with simulation, but the best way to guarantee an optimum test set is to perform *diagnostic test pattern generation* (DTPG). A DTPG system considers a pair of faults that are not distinguished by the present test set; it then either finds a test that distinguishes between the two faults or proves that there is no test that distinguishes between them.

After the fault list and test set are selected, the faults are simulated against the tests, and the resulting fault simulation data must be stored in a space-efficient manner. Dictionary compaction schemes are lossy: They attempt to remove information from the dictionary without altering the sizes of the equivalence classes contained in the dictionary. If only faults contained in the dictionary are considered, the dictionary is equally useful before and after compaction. But what if the chip to be diagnosed contains a fault that is not in the dictionary? If the fault dictionary undergoes compaction, information important to the diagnosis of unmodeled faults may be lost.

I begin in Chapter 2 by discussing existing diagnostic test pattern generation systems. I then present a diagnostic test pattern generator for stuck-at and $I_{DDQ}$ detectable bridging faults. In Chapter 3 I look at common dictionary compaction techniques, introduce a new method of lossless dictionary compression, and show how the new method can be combined with existing techniques. In Section 4 I apply a variety of dictionary types to the problem of diagnosing unmodeled faults and show that common metrics for evaluating dictionary quality are inadequate for determining a dictionary's ability to diagnose faults it does not contain.

# 2. Diagnostic Test Pattern Generation

Being able to generate a test that discriminates between two faults (or prove that there is no such test because the faults will always be in the same equivalence class) will help build more effective fault dictionaries. A simulation-based approach to dictionary construction is incapable of determining that two faults are truly indistinguishable; the simulator may continue to evaluate new random patterns in the fruitless hope of increasing the diagnostic capability of the dictionary. Diagnostic test pattern generation can put an end to the search for a differentiating test when none exists.

Camurati *et al.* [8] presented the $\Delta$-algorithm, a preliminary version of a PODEM-based approach [13], which generates diagnostic test patterns for stuck-at faults. They made the important observation that DTPG bears a strong resemblance to traditional automatic test pattern generation (ATPG). ATPG involves two circuits: the fault-free circuit and the faulty circuit. Diagnostic test generation also involves two circuits, but instead of a fault-free circuit, the test generator examines a second faulty circuit. A test that distinguishes between the two faults will cause the outputs of the circuits to differ. The $\Delta$-algorithm also introduced the idea that a distinguishing test must create a path from one of the fault sites to a primary circuit output along which the two faulty circuits will carry different values. This idea is similar to the active path heuristic discussed in the following section. The $\Delta$-algorithm met with limited success: It was able to process stuck-at faults only on fanout-stems and was useful on only smaller benchmark circuits.

Grüning *et al.* [14] presented DIATEST, a more complete diagnostic test pattern generator for stuck-at faults based on the CONTEST ATPG system [18]. Like the $\Delta$-algorithm,

DIATEST attempts to generate a test that distinguishes between two faults by replacing the fault-free circuit used in ATPG with a second faulty circuit. Similarly, DIATEST makes use of the fact that at least one of the faults must be detected. If neither fault is detected then the outputs of both of the faulty circuits are identical, and the faults are not distinguished. DIATEST makes use of dynamic learning as proposed by Schulz *et al.* [27] and is very successful, producing no aborted fault pairs for the ISCAS-85 circuits [7].

Nemesis, UCSC's test pattern generator [17], generates a test for a given fault by extracting a formula in conjunctive normal form (CNF) and applying a Boolean satisfier to the resulting formula. The same satisfier is used for all types of test pattern generation, but the manner in which a formula is extracted changes depending on the fault model and the type of test desired. For this thesis, rather than display the formulas in CNF (which many find unintuitive), I will show the equalities that the CNF is used to represent. That is, in order to represent the constraints imposed by an AND gate I will give the equality $(\mathrm{C} = \mathrm{A} \cdot \mathrm{B})$ rather than the conjunctions $(\overline{\mathrm{A}} + \overline{\mathrm{B}} + \mathrm{C}) \cdot (\mathrm{A} + \overline{\mathrm{C}}) \cdot (\mathrm{B} + \overline{\mathrm{C}})$. Both of these representations contain the implications $(\mathrm{C} \rightarrow \mathrm{A} \cdot \mathrm{B})$ and $(\mathrm{A} \cdot \mathrm{B} \rightarrow \mathrm{C})$. That is, if A=1 and B=1 then it must be true that C=1, and if A=1 and C=0 then it must be true that B=0.

It is important to note that for combinational circuits the only values encountered during fault simulation and test generation are logic 0 and logic 1. Because there are no unknowns, faults are partitioned by an equivalence relation: If fault A is equivalent to fault B, and fault B is equivalent to fault C, then fault A is also equivalent to fault C [16, 24]. This allows us to prove all the members of a large equivalence class indistinguishable without considering every possible pair of faults in the class. In other words, in order to determine

that there is an equivalence class of size eight that cannot be broken into smaller equivalence classes, Nemesis need only to prove that each of the first seven faults in indistinguishable from the eighth. This requires only seven attempts at diagnostic test pattern generation and not $\binom{8}{2}$, or 28, attempts at diagnostic test pattern generation.

## 2.1 DTPG for stuck-at faults

Although many realistic circuit defects cannot be modeled as single stuck-at faults, many can. Additionally, testing for single stuck-at faults is useful because a complete single stuck-at test set for an irredundant circuit stimulates and makes observable every portion of the circuit. How do stuck-at faults perform when the realm of interest is diagnosis and not detection? Ideally, stuck-at faults will guide the failure analysis engineer to the correct area: He or she can then examine a small physical region of the circuit to identify the actual defect. Even though faulty chips do not always contain stuck-at faults, the better the resolution of the stuck-at dictionary, the easier the job of physical defect location.

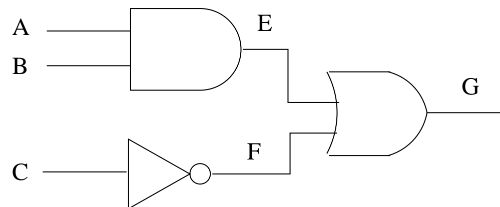


Figure 2.1: A small example circuit.

Figure 2.1 shows a simple circuit that I will use to illustrate how Nemesis generates a test that detects a single stuck-at fault or distinguishes between two single stuck-at faults.

A test that detects a stuck-at fault must cause the outputs of the fault-free circuit and the faulty circuit to differ. For a circuit with a single output, this is equivalent to applying an

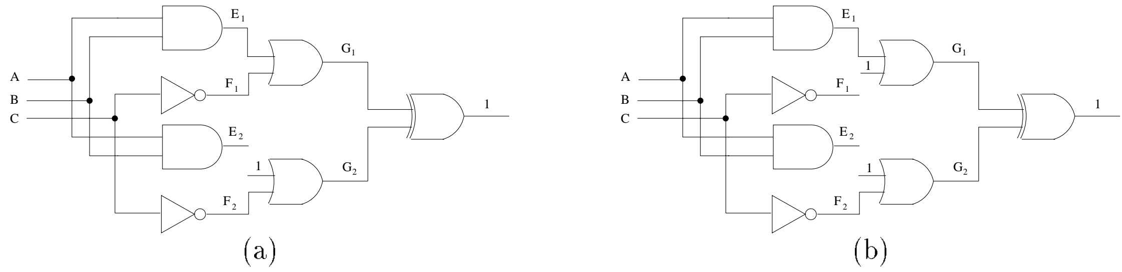


Figure 2.2: Testing for E stuck-at 1 and distinguishing between E and F stuck-at 1.

exclusive-or (XOR) to the outputs of the fault-free and faulty circuits and requiring that the output of the XOR be a logic 1. This concept can easily be extended to circuits with multiple outputs. Figure 2.2a illustrates how Nemesis generates a test for node E stuck-at 1. (This example shows the fault-free and faulty circuits separately, but the implementation does not duplicate any gates not in the fault propagation cone.) Nemesis extracts the formula for the fault-free circuit, the formula for the faulty circuit, and the formula requiring that the outputs of the two circuits must differ [17]. The resulting formula is

$$(\mathrm{E}_1 = \mathrm{A} \cdot \mathrm{B}) \cdot (\mathrm{F}_1 = \overline{\mathrm{C}}) \cdot (\mathrm{G}_1 = \mathrm{E}_1 + \mathrm{F}_1) \cdot$$
$$(\mathrm{E}_2 = 0 = \mathrm{A} \cdot \mathrm{B}) \cdot (\mathrm{F}_2 = \overline{\mathrm{C}}) \cdot (\mathrm{G}_2 = 1 + \mathrm{F}_2) \cdot (\mathrm{G}_1 \neq \mathrm{G}_2).$$

Any variable assignment that satisfies this formula (sets it to 1) must be a test for E stuck-at 1. If no such assignment exists, there is no test for the fault. One satisfying assignment is ABC = 101.

In order to generate a test that distinguishes between two stuck-at faults, Nemesis requires that the outputs of the two faulty circuits differ. Again, this constraint can be viewed as an XOR of the two faulty circuit outputs with the output of the XOR set to logic 1. Figure 2.2b shows how Nemesis generates a test that distinguishes between E stuck-

at 1 and F stuck-at 1. It extracts the formula for the first faulty circuit, the formula for the second faulty circuit, and a formula requiring that the outputs of the two circuits must differ. The resulting formula is

$$(\mathrm{E}_1 = \mathrm{A} \cdot \mathrm{B}) \cdot (\mathrm{F}_1 = \overline{\mathrm{C}}) \cdot (\mathrm{G}_1 = \mathrm{E}_1 + 1) \cdot (\mathrm{E}_2 = \mathrm{A} \cdot \mathrm{B}) \cdot (\mathrm{F}_2 = \overline{\mathrm{C}}) \cdot$$
$$(\mathrm{G}_2 = 1 + \mathrm{F}_2) \cdot (\mathrm{G}_1 \neq \mathrm{G}_2).$$

If no variable assignment exists that satisfies this formula, there is no test that distinguishes between E stuck-at 1 and F stuck-at 1. Any variable assignment that satisfies the formula must be a test that distinguishes between the two faults. This test may detect the first fault, the second fault, or both faults (on different primary outputs), but it must always detect at least one of the two faults (otherwise they would not be distinguished and the formula would not have been satisfied). This formula has no satisfying assignment, which is not a surprise—stuck-at 1 faults on the inputs of an OR gate are always in an equivalence class along with the output of the OR gate stuck-at 1. Normally these faults would not be targeted for diagnostic test generation; the equivalence class would be identified by a fault collapsing routine before DTPG.

As described in a previous publication [17], Nemesis uses the *active path heuristic* to make test pattern generation more efficient. For single stuck-at test generation, an active path is a path from the fault site to a primary output along which an error is propagated. Nemesis adds constraints to the formula that require there to be at least one active path present for test generation to be successful. Since a test for a stuck-at fault must make the effect of the fault be visible on at least one primary output, active path information is redundant, but this variety of redundant information dramatically improves the performance

of the test pattern generator.

In order to apply the active path heuristic to diagnostic test pattern generation, two modifications are needed:

1. Instead of a path of errors, Nemesis requires a path of differences. Since there is no fault-free circuit, the concept of an error does not apply to distinguishing between the two faulty circuits.

2. There is no longer a single site for the beginning of the active path: either fault (or both faults) may begin an active path.

These distinctions required the reimplementation of the portion of Nemesis that added active path constraints to the formula. When an active path moves through both fault sites, creating the correct constraints to force the output of the two faulty circuits to differ requires particular care: An active path from one fault site can not pass through discrepancies generated by the other fault site. If both faults place a discrepancy on the same wire, that wire cannot be used to distinguish between the two faults, and hence the wire cannot be on an active path.

## 2.2 Stuck-at DTPG results

Nemesis simulates a previously generated complete and compacted stuck-at test set against all stuck-at faults. Based on the simulation results, Nemesis removes all undetectable faults from the fault list and computes the equivalence classes for the remaining faults. It then targets all multiple-fault equivalence classes for DTPG. (As mentioned at the beginning of the chapter, it is not necessary to target all possible pairs of faults in an equivalence class

in order to prove that the equivalence class cannot be decomposed further.) Whenever the system generates a distinguishing test, all faults are simulated against the new test and equivalence classes are recomputed. Nemesis continues to process equivalence classes until all faults have been either uniquely distinguished or proven to be an inseparable member of an equivalence class.

Table 2.1 shows Nemesis' performance on the ISCAS-85 benchmark circuits [7], listing the total number of stuck-at faults, the number of fault pairs targeted by the DTPG engine, and its performance when it targeted these fault pairs. The results include the number of distinguishing tests that were generated, the number of pairs that were proved indistinguishable, and the time it took to process all pairs (Nemesis did not fail to process any fault pair). The times reported include simulation and equivalence class computation as well as DTPG on a Digital Equipment Corporation Alpha 3000/600.

| Circuit | stuck-at | pairs | fault-pairs | | | time |
|---|---|---|---|---|---|---|
| | faults | targeted | tested | proved | failed | (secs) |
| C432 | 524 | 22 | 9 | 13 | 0 | 15 |
| C499 | 758 | 14 | 2 | 12 | 0 | 4 |
| C880 | 942 | 61 | 6 | 55 | 0 | 6 |
| C1355 | 1574 | 636 | 0 | 636 | 0 | 127 |
| C1908 | 1879 | 270 | 19 | 251 | 0 | 89 |
| C2670 | 2747 | 374 | 39 | 335 | 0 | 206 |
| C3540 | 3428 | 362 | 28 | 334 | 0 | 217 |
| C5315 | 5350 | 459 | 46 | 413 | 0 | 281 |
| C6288 | 7744 | 1027 | 16 | 1011 | 0 | 788 |
| C7552 | 7550 | 1011 | 46 | 965 | 0 | 1198 |

Table 2.1: Diagnostic Test Pattern Generation for stuck-at faults.

For the larger circuits, the number of targeted pairs is four to thirty times greater than the number of faults that have to be directly targeted for normal stuck-at test pattern generation. The concern that this number could explode as the circuit sizes continue to increase can be mitigated by considering the option to perform diagnostic test pattern

| Circuit | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C0432 | 473 | 22 | 1 | | | | | | | | |
| C0499 | 724 | 10 | 2 | | | | | | | | |
| C0880 | 820 | 61 | | | | | | | | | |
| C1355 | 398 | 428 | 104 | | | | | | | | |
| C1908 | 1325 | 257 | 5 | | | | | 2 | | | |
| C2670 | 1908 | 202 | 68 | 14 | 3 | 1 | | 1 | 2 | | 1 |
| C3540 | 2650 | 206 | 35 | 16 | | 6 | | 3 | | | |
| C5315 | 4383 | 384 | 41 | 3 | 1 | | | | | | |
| C6288 | 5602 | 1033 | 14 | | | | | | | | |
| C7552 | 5529 | 704 | 105 | 9 | 8 | | 1 | | | 1 | |

Table 2.2: Number of stuck-at fault equivalence classes by size before DTPG.

| Circuit | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C0432 | 494 | 13 | | | | | | | | | |
| C0499 | 726 | 12 | | | | | | | | | |
| C0880 | 832 | 55 | | | | | | | | | |
| C1355 | 398 | 428 | 104 | | | | | | | | |
| C1908 | 1382 | 233 | 2 | | | | | 2 | | | |
| C2670 | 2035 | 204 | 49 | 4 | 1 | | | 1 | | | 1 |
| C3540 | 2726 | 174 | 35 | 13 | | 6 | | 3 | | | |
| C5315 | 4497 | 351 | 28 | 2 | | | | | | | |
| C6288 | 5690 | 1007 | 2 | | | | | | | | |
| C7552 | 5612 | 737 | 96 | 2 | 6 | | 1 | | | | |

Table 2.3: Number of stuck-at fault equivalence classes by size after DTPG.

generation as part of a two-phase diagnostic procedure [25] where only the equivalence class identified by a cursory initial diagnosis would be subject to DTPG.

The precise effect of DTPG can be evaluated by examining the sizes of the fault equivalence classes. The sizes of all equivalence classes before and after DTPG are given in Tables 2.2 and 2.3. A metric that gives a more concise view of this information is *diagnostic expectation*. The diagnostic expectation for a fault list and test set gives the average equivalence class size over all faults [26]. A diagnostic expectation of 1 would indicate that every fault could be uniquely identified. A diagnostic expectation of 10 would indicate that, on average, there will be an equivalence class of size 10 after diagnosing a faulty circuit. Table 2.4 shows the diagnostic expectation for the detectable faults before and after DTPG.

| Circuit | Before | After |
|---|---|---|
| C432 | 1.10 | 1.05 |
| C499 | 1.04 | 1.03 |
| C880 | 1.13 | 1.12 |
| C1355 | 1.95 | 1.95 |
| C1908 | 1.34 | 1.32 |
| C2670 | 1.52 | 1.36 |
| C3540 | 1.35 | 1.32 |
| C5315 | 1.20 | 1.17 |
| C6288 | 1.28 | 1.26 |
| C7552 | 1.34 | 1.30 |

Table 2.4: Diagnostic expectation before and after stuck-at DTPG

Diagnostic Test pattern generation shows that the diagnostic expectation of the initial test sets is close to the optimum value obtainable. DTPG is worthwhile because it has proved that the initial test sets were good; it has put an end to random simulation in an attempt to further improve the dictionary. Nemesis is the first system to report the minimum sizes of all of the equivalence classes for the ISCAS-85 benchmarks.

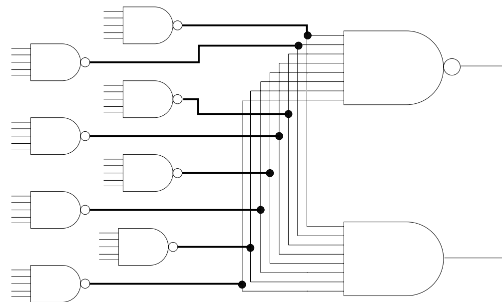

Figure 2.3: The 8 stuck-at 0 faults on the fanout stems are indistinguishable

Figure 2.3 shows a size-8 equivalence class from the C1908 (a stuck-at 0 fault on each of the fanout stems is indistinguishable from every other such fault). There are two appearances of this subcircuit in the C1908, and they are the largest equivalence classes in the circuit. This equivalence class is symmetric and simple, but I did not find it until I examined the DTPG results. This experience indicates that new methods of fault collapsing may be suggested by analyzing DTPG output.

## 2.3 DTPG for $I_{DDQ}$ detectable bridging faults

$I_{DDQ}$ testing is a powerful technique, but the application of $I_{DDQ}$ tests is much more time consuming than the application of standard voltage tests; after the application of a test, the tester cannot accurately measure the quiescent current of the chip under test until after the chip's transient current consumption has died away. For this reason, the number of $I_{DDQ}$ tests applied during production testing is usually very small. For diagnosis, the amount of time the chip spends on the tester is not as important, but test engineers frequently use the same test set. A very short $I_{DDQ}$ test set created for identifying faulty parts may not do a good job of diagnosing them. In order to improve the initial test sets, I extended the Nemesis ATPG system to generate diagnostic tests for realistic bridging faults using $I_{DDQ}$ measurement.

A bridging fault that creates a feedback loop can cause an otherwise combinational circuit to oscillate or hold state. Although feedback is an important issue for determining the logic behavior of bridging faults, for the purposes of $I_{DDQ}$ testing, any test that forces two wires to opposite Boolean values in the fault free circuit will create a detectable amount of excess $I_{DDQ}$ current in a circuit containing a bridge between the two wires. If a test causes oscillation, gates will continue to switch long after they should have settled to a final value (thus causing excess current); If a test causes sequential behavior, the circuit will contain a static conducting path from power to ground.

In order for a test to differentiate between two bridging faults, the test must detect one of the two bridges but not both. For the bridging faults pictured in Figure 2.4, Nemesis adds the following constraints to the formula that represents the fault-free behavior of the

circuit:

$$(X = (a \neq b)) \cdot (Y = (c \neq d)) \cdot (X \neq Y).$$

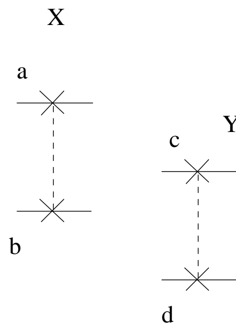


Figure 2.4: A differentiating $I_{DDQ}$ test detects X or Y but not both.

## 2.4 Bridge $I_{DDQ}$ DTPG results

Table 2.5 shows the system's performance on the MCNC versions of the ISCAS-85 benchmark circuits. The bridging fault lists were extracted from the circuit layouts by Carafe, an inductive fault analysis tool [15]. Nemesis simulates a complete $I_{DDQ}$ test set for all detectable bridging faults in the circuit. Based on simulation results, undetectable faults are removed from the faultlist, and Nemesis partitions the remaining faults into equivalence classes. All multiple fault equivalence classes are then targeted for DTPG.

Since there are many more realistic bridging faults than stuck-at faults, Nemesis is forced to target many more fault pairs than it does in diagnostic DTPG for stuck-at faults. The increased workload affects the number of failed faults, but CPU time is not unacceptably high.

Especially for small circuits, diagnostic test pattern generation noticeably improves diagnostic expectation. Table 2.6 shows the diagnostic expectation for the complete $I_{DDQ}$

| Circuit | bridge faults | pairs targeted | fault-pairs tested | proved | failed | time (secs) |
|---|---|---|---|---|---|---|
| C432 | 1595 | 112 | 11 | 101 | 0 | 2 |
| C499 | 2781 | 91 | 17 | 74 | 0 | 4 |
| C880 | 3275 | 147 | 10 | 137 | 0 | 3 |
| C1355 | 4422 | 263 | 50 | 213 | 0 | 18 |
| C1908 | 4744 | 550 | 17 | 522 | 11 | 29 |
| C2670 | 13710 | 1746 | 20 | 1663 | 63 | 801 |
| C3540 | 16459 | 1796 | 29 | 1767 | 0 | 92 |
| C5315 | 40430 | 3546 | 28 | 3518 | 0 | 281 |
| C6288 | 21911 | 1297 | 18 | 1279 | 0 | 104 |
| C7552 | 53789 | 7618 | 30 | 7468 | 120 | 2115 |

Table 2.5: DTPG for $I_{DDQ}$ detectable bridging faults.

detection test set and the DTPG test set. DTPG also greatly increases the number of tests in the test set (Table 2.7). In order to isolate the effect of increased test set size on diagnostic expectation, I augmented a complete $I_{DDQ}$ test set with enough random tests to bring it to the same size as the DTPG test set. The results for the augmented test set appear in Table 2.6. As expected, the random augmented test set performs significantly better than the complete test set, but not as well as the DTPG test set. In contrast to complete stuck-at fault test sets, test sets for detecting bridging faults with $I_{DDQ}$ measurement greatly benefit from DTPG.

| Circuit | Complete | DTPG | Augmented |
|---|---|---|---|
| C432 | 2.77 | 1.64 | 1.81 |
| C499 | 1.30 | 1.08 | 1.19 |
| C880 | 2.07 | 1.56 | 1.68 |
| C1355 | 3.50 | 1.80 | 2.06 |
| C1908 | 1.75 | 1.49 | 1.59 |
| C2670 | 3.09 | 2.59 | 2.70 |
| C3540 | 2.55 | 2.35 | 2.43 |
| C5315 | 2.54 | 2.39 | 2.39 |
| C6288 | 5.53 | 5.05 | 5.13 |
| C7552 | 3.44 | 3.26 | 3.35 |

Table 2.6: Diagnostic expectation for the $I_{DDQ}$ test sets.

Table 2.8, 2.9, and 2.10 give the equivalence class sizes for each circuit for the complete test set, the test set after DTPG, and the test set augmented with random patterns. The

| Circuit | Complete | DTPG | Augmented |
|---|---|---|---|
| C432 | 13 | 24 | 24 |
| C499 | 20 | 37 | 37 |
| C880 | 15 | 25 | 25 |
| C1355 | 16 | 66 | 66 |
| C1908 | 25 | 42 | 42 |
| C2670 | 16 | 36 | 36 |
| C3540 | 29 | 58 | 58 |
| C5315 | 22 | 50 | 50 |
| C6288 | 30 | 48 | 48 |
| C7552 | 34 | 64 | 64 |

Table 2.7: Test set size.

largest equivalence class for each of the circuits is comprised of faults that are detected by every test. The size of this equivalence class for each of the three test sets is shown in Table 2.11. The size of the largest equivalence class is minimized by the DTPG process. The faulty circuit in Figure 2.5 will draw excess $I_{DDQ}$ current for 97% of randomly applied inputs, but tests that do not cause excess $I_{DDQ}$ current do exist. DTPG will differentiate between faulty circuits that always cause excess $I_{DDQ}$ current and those that almost always cause excess $I_{DDQ}$ current.

| Circuit | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | large |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C0432 | 984 | 179 | 35 | 17 | 3 | 3 | | | | | 1 |
| C0499 | 2225 | 222 | 27 | 2 | 2 | | | | | | 1 |
| C0880 | 2529 | 260 | 39 | 7 | 6 | | | | | | 1 |
| C1355 | 2954 | 358 | 105 | 37 | 15 | 9 | 4 | | 2 | 2 | 1 |
| C1908 | 3359 | 464 | 88 | 20 | 8 | 2 | 1 | 1 | | | 1 |
| C2670 | 8406 | 1690 | 370 | 94 | 33 | 5 | 7 | 3 | | | 1 |
| C3540 | 12628 | 1552 | 133 | 26 | 7 | 2 | | | | | 2 |
| C5315 | 32096 | 2976 | 408 | 114 | 49 | 15 | 7 | 4 | 1 | 1 | 2 |
| C6288 | 19215 | 1106 | 32 | 6 | 1 | 3 | 2 | | 1 | | 1 |
| C7552 | 39315 | 4522 | 1064 | 309 | 64 | 31 | 10 | 3 | 1 | 1 | 1 |

Table 2.8: Equivalence classes for $I_{DDQ}$ detectable bridging faults by size before DTPG.

| Circuit | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | large |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C0432 | 1429 | 55 | 4 | 1 | | | | 1 | | | 1 |
| C0499 | 2646 | 55 | 5 | | | | | | | 1 | |
| C0880 | 3043 | 91 | 3 | | | | | | | | 1 |
| C1355 | 4059 | 135 | 11 | | | | | | | | 1 |
| C1908 | 3758 | 403 | 34 | 8 | 1 | | | | | | 1 |
| C2670 | 10647 | 1079 | 190 | 32 | 8 | | 1 | | | | 1 |
| C3540 | 13175 | 1386 | 88 | 16 | 4 | 1 | | | | | 1 |
| C5315 | 34019 | 2551 | 235 | 55 | 17 | 2 | 1 | 1 | 1 | | 2 |
| C6288 | 19645 | 973 | 6 | | | | | | | | 1 |
| C7552 | 40522 | 4382 | 941 | 241 | 42 | 12 | 4 | 3 | 1 | | 1 |

Table 2.9: Equivalence classes for $I_{DDQ}$ detectable bridging faults by size after DTPG.

| Circuit | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | large |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C0432 | 1324 | 98 | 9 | 1 | | | | | | 1 | 1 |
| C0499 | 2394 | 168 | 12 | 1 | | | | | | | 1 |
| C0880 | 2989 | 116 | 3 | | | | | | | | 1 |
| C1355 | 3939 | 155 | 31 | 3 | | | | | | | 1 |
| C1908 | 3605 | 430 | 56 | 10 | 3 | 1 | 1 | | | | 1 |
| C2670 | 10588 | 1090 | 193 | 35 | 9 | 1 | 1 | | | | 1 |
| C3540 | 13005 | 1433 | 109 | 17 | 5 | 1 | | | | | 1 |
| C5315 | 33951 | 2554 | 249 | 60 | 17 | 2 | 1 | 1 | 1 | | 2 |
| C6288 | 19587 | 997 | 7 | 1 | | | | | | | 1 |
| C7552 | 39946 | 4415 | 1005 | 285 | 55 | 20 | 6 | 3 | 1 | 1 | 1 |

Table 2.10: Equivalence classes for $I_{DDQ}$ detectable bridging faults by size with a complete $I_{DDQ}$ test set augmented with random patterns.

| Circuit | Complete | DTPG | Augmented |
|---|---|---|---|
| C432 | 44 | 29 | 31 |
| C499 | 13 | 10 | 11 |
| C880 | 51 | 41 | 45 |
| C1355 | 91 | 57 | 65 |
| C1908 | 40 | 35 | 37 |
| C2670 | 144 | 134 | 139 |
| C3540 | 145 | 137 | 141 |
| C5315 | 224 | 220 | 220 |
| C6288 | 311 | 295 | 298 |
| C7552 | 331 | 321 | 326 |

Table 2.11: Number of faults that are detected by every $I_{DDQ}$ test.

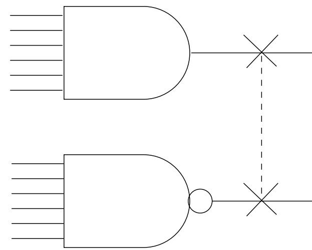

Figure 2.5: A bridging fault that will cause excess $I_{DDQ}$ current for 97% of randomly applied tests.

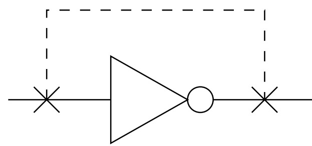

Figure 2.6: A bridging fault that will always cause excess $I_{DDQ}$ current.

# 3. Dictionary organization

In this chapter I present a new lossless dictionary format and compare the new organization with previously published lossy dictionary schemes. After reviewing several methods of dictionary organization, I present the the concept of the *error set*, around which the new organization is built. Having done that, I explain the new organization and how it can be used in conjunction with other compaction schemes.

## 3.1 Popular dictionary organizations

As a prelude to describing the new dictionary organization, I briefly discuss previous work on the subject of dictionary compaction. The C17 (shown in Figure 3.1) will serve as an example circuit. To keep the example small, only the eight faults listed in Table 3.1 and the four tests in Table 3.1 will be considered.

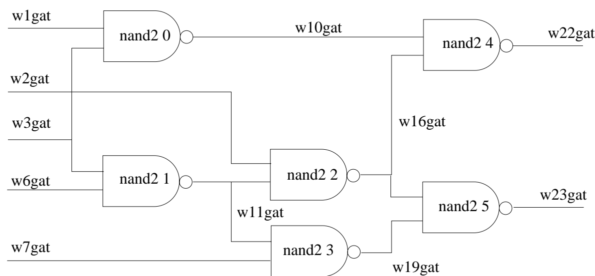


Figure 3.1: The C17.

The *full response dictionary*, which stores the circuit outputs in the presence of each fault for each test, appears in Table 3.3. This dictionary has the advantage of providing all the information available for a given test set. Further, it can easily be stored in a bit-packed manner. A *pass/fail* dictionary (Table 3.4) is similar to the full dictionary but only stores

| Fault # | Fault Name |
| --- | --- |
| 1 | primary input w3gat stuck at 1 |
| 2 | input pin 1 of nand2 1 stuck at 1 |
| 3 | output pin of nand2 1 stuck at 0 |
| 4 | output pin of nand2 2 stuck at 1 |
| 5 | input pin 2 of nand2 4 stuck at 1 |
| 6 | output pin of nand2 4 stuck at 0 |
| 7 | output pin of nand2 5 stuck at 0 |
| 8 | primary input w2gat stuck at 1 |

Table 3.1: Partial fault list

| Test | Inputs | Outputs |
| --- | --- | --- |
| 1 | 11011 | 11 |
| 2 | 10000 | 00 |
| 3 | 11000 | 11 |
| 4 | 10011 | 01 |

Table 3.2: Test set

| Fault | Test | | | |
| --- | --- | --- | --- | --- |
| # | 1 | 2 | 3 | 4 |
| 1 | 10 | 10 | 11 | 10 |
| 2 | 00 | 00 | 11 | 00 |
| 3 | 00 | 00 | 00 | 00 |
| 4 | 01 | 00 | 00 | 01 |
| 5 | 01 | 00 | 01 | 01 |
| 6 | 01 | 00 | 01 | 01 |
| 7 | 10 | 00 | 10 | 00 |
| 8 | 11 | 11 | 11 | 11 |

Table 3.3: Full response dictionary

| Fault | Test | | | |
| --- | --- | --- | --- | --- |
| # | 1 | 2 | 3 | 4 |
| 1 | 1 | 1 | 0 | 1 |
| 2 | 1 | 0 | 0 | 1 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 0 | 1 | 0 |
| 5 | 1 | 0 | 1 | 0 |
| 6 | 1 | 0 | 1 | 0 |
| 7 | 1 | 0 | 1 | 1 |
| 8 | 0 | 1 | 0 | 1 |

Table 3.4: Pass/Fail dictionary

one bit of information per fault-vector pair—a 1 if the circuit will fail the test in the presence of the fault and a 0 if it will pass. The diagnostic expectation of the pass/fail dictionary is worse than the diagnostic expectation for the full dictionary, but if the circuit has a large number of primary outputs, the pass/fail dictionary is significantly smaller.

Pomeranz and Reddy's *Compact* dictionary uses a greedy algorithm to choose columns from the full response dictionary to augment a pass/fail dictionary [21]. The result is a compacted dictionary with little or no erosion of diagnostic expectation (for the modeled faults). The Compact dictionary in Table 3.5 is made up of the pass/fail dictionary and two columns from the full dictionary: output 1 for vector 1 and output 2 for vector 3. It contains the same equivalence classes as the full dictionary.

Another popular dictionary style is the detection dictionary, shown in Table 3.6. This dictionary stores only the output patterns with errors for each fault. Since it is unlikely

| Fault # | Test 1 | | 2 | 3 | | 4 |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 2 | 1 | 0 | 0 | 0 | 1 | 1 |
| 3 | 1 | 0 | 0 | 1 | 0 | 1 |
| 4 | 1 | 0 | 0 | 1 | 0 | 0 |
| 5 | 1 | 0 | 0 | 1 | 1 | 0 |
| 6 | 1 | 0 | 0 | 1 | 1 | 0 |
| 7 | 1 | 1 | 0 | 1 | 0 | 1 |
| 8 | 0 | 1 | 1 | 0 | 1 | 1 |

Table 3.5: Compact dictionary

that all faults will be detected by all tests, fewer output sequences need to be stored in the detection dictionary than in the full response dictionary. The detection dictionary is easy to compact—a drop-on-K dictionary is a detection dictionary where the number of detections for each fault is limited to K. The special case K=1 is known as a *vector* dictionary (Table 3.7). A vector dictionary stores the same information as the pass/fail dictionary. Unlike the full response dictionary, a detection dictionary cannot be stored in a bit-packed format;the full response dictionary's regular array of of ones and zeros is now irregular and interspersed with integers.

| Fault # | Detections |
|---|---|
| 1 | 1:10 2:10 4:10 |
| 2 | 1:00 4:00 |
| 3 | 1:00 3:00 4:00 |
| 4 | 1:00 3:00 |
| 5 | 1:01 3:01 |
| 6 | 1:01 3:01 |
| 7 | 1:10 3:10 4:00 |
| 8 | 2:11 4:11 |

Table 3.6: Detection dictionary

| Fault # | Detections |
|---|---|
| 1 | 1 2 4 |
| 2 | 1 4 |
| 3 | 1 3 4 |
| 4 | 1 3 |
| 5 | 1 3 |
| 6 | 1 3 |
| 7 | 1 3 4 |
| 8 | 2 4 |

Table 3.7: Vector dictionary

Boppana and Fuchs introduced two methods for compacting fault dictionaries based on *diagnostic trees* [1, 5]. A diagnostic tree gives the equivalence classes that are distinguished after each test is applied. The diagnostic tree for the example appears in Figure 3.2.

Both of the proposed methods use a tree-based analysis to remove information from the dictionary that does not contribute to increased diagnostic performance. For example, information about fault 1 does not need to be recorded after the second vector; the fault is already uniquely distinguished [1]. After the diagnostic tree is compacted it is stored in a set of tables. Compaction method $DC1$ stores the tree in two tables; the first stores an entry for the tree nodes. Each node contains the name of its parent node and the output sequence leading from the parent node. The second table contains the name of the leaf node corresponding to each fault. Compaction method $DC2$ adds a third table—a fault-vector matrix that contains a 1 whenever a vector helps to further distinguish a fault. The addition of this table reduces the potential size of each node entry in the first table.

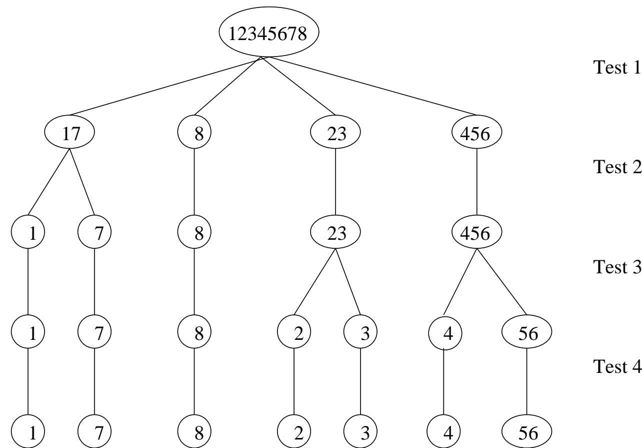


Figure 3.2: The diagnostic tree

## 3.2 The Error Set

In order to retain all diagnostic information available, a dictionary must store the response of every fault to every test. The full dictionary achieves this goal in an inefficient manner; many entries are identical to the fault-free output. The detection dictionary

overcomes this deficiency by only storing the output patterns for which the fault is detected, but this representation is also inefficient: Many of the bits in each output pattern do not carry discrepancies, and the same faulty output pattern may be repeated many times. The optimal dictionary will only store the difference between the fault-free circuit and each of the faulty circuits.

To achieve this goal, our dictionary format is based on *error sets*. An error set is a set of primary outputs that carry errors. As such, error sets are not specific to input patterns, output patterns, or faults. In order to create a specific faulty output pattern using an error set, the error set is simply associated with a fault-free output pattern. The faulty output pattern is then the fault-free output pattern with inverted values in the positions specified by the error set.

As an example of how one error set can be used for different patterns and different faults, consider the full response dictionary in Table 3.3. Test 1 will cause the circuit containing fault 2 to have errors on both outputs (the fault-free response is 11 and the faulty response is 00). Similarly, test 2 will cause the circuit containing fault 8 to have errors on both outputs (the fault-free response is 00 and the faulty response is 11), and test 4 will cause the circuit containing fault 1 to have errors on both outputs (the fault-free response is 01 and the faulty response is 10). All three fault/test pairs provoke the same error set[1].

My motivation for the new dictionary organization using error sets is threefold:

1. Frequently a fault causes the same error set for many vectors.

---

[1]Error sets can be compared to the *POPAT*s used by Teradyne's LASAR system [22]. A POPAT is a pairing of an input pattern (a test) and a primary output that will carry an observable error when that input pattern is applied. The one error set I discussed in the above example would be entered as several different POPATs in Teradyne's dictionary.

2. Most faults create only a handful of different error sets.

3. The same error set is provoked by many different faults.

In theory, the number of different error sets could be as large as the minimum of two upper bounds: the first upper bound is $2^n$ (where n is the number of primary outputs) and second upper bound is the number of tests multiplied by the number of faults. In practice, however, I have found the number of error sets is much smaller than the worst-case upper bound. I will report on the number of error sets for the ISCAS circuits in Section 3.6.

### 3.3 Dictionary organization using error sets

As noted earlier, a particular faulty output pattern is specified completely by the combination of a fault-free output pattern and an error set. Having previously listed all the input patterns and their fault-free responses, a dictionary could store the complete response of a fault to the test set by storing vector/error set pairs. But because error sets are reused so often, the basic entry for a fault in the error set dictionary is an error set followed by a list of vectors that provoke that error set. There may be many error set entries per fault. As a final detail, instead of storing actual error sets or vectors, the dictionary stores only indexes into an error set table and a vector table.

Table 3.8 shows the two tables that comprise the error set dictionary. Table 3.8a shows the three error sets and Table 3.8b shows the responses of the eight faults. Table 3.8b looks like a matrix, but the entries for each fault are made up of error set/vector list pairs; accessing the error set dictionary is similar to accessing the detection dictionary. The error set dictionary in Table 3.8 contains all the diagnostic information of the full response

| Error set | Faulty Outputs | Bit Mask |
|---|---|---|
| 1 | w23gat | 01 |
| 2 | w22gat w23gat | 11 |
| 3 | w22gat | 10 |

(a)

| Fault | Error set | | |
|---|---|---|---|
| # | 1 | 2 | 3 |
| 1 | 1 | 4 | 2 |
| 2 | 4 | 1 | |
| 3 | 4 | 1 3 | |
| 4 | | 3 | 1 |
| 5 | | | 1 3 |
| 6 | | | 1 3 |
| 7 | 1 3 4 | | |
| 8 | | 2 | 4 |

(b)

Table 3.8: Error sets and final dictionary organization

dictionary. As a further optimization, if a complete vector sequence appears for a single fault in a single error set, it can be replaced by an elided sequence. That is, 2, 3, 4, 5 can be replaced by 2–5. Because of this optimization, the dictionary may contain more fault detections than data items.

The error set dictionary format incorporates unknowns by extending the definition of an error set: a special symbol is used as a prefix to distinguish outputs that carry unknowns from outputs that carry errors. Thus, the error set dictionary has no problem recording information for circuits that lack reset states or with fault models that generate unknowns such as bridging or break faults.

## 3.4 Further dictionary compaction using error sets

One of the primary motivations for the dictionary format using error sets is the desire to include all diagnostic information. I expand on the importance of including all diagnostic information in Chapter 4.

In order to show that the new organization is competitive with lossy formats, I give an

example of how the organization can be combined with compaction methods. Since the error set format is detection-based, it can easily make use of schemes that limit the number of detections recorded for each fault. Drop-on-K is one possibility, but it may limit the diagnostic capability of the dictionary. In order to further compact the new dictionaries without losing any diagnostic capability for the modeled faults, the simulator can drop faults after they have been completely distinguished [1, 5]. This requires that the system maintain fault equivalence classes during simulation, but in return the dictionary provides the same diagnostic expectation as a full dictionary without requiring the simulation of every fault against every test. The result is a significant speedup of fault simulation and an impressive reduction in dictionary size. I refer to dictionaries created using this technique as *drop-when-distinguished* dictionaries.

Table 3.9 shows the error set dictionary resulting from dropping fully distinguished faults for the example. Note that fault 8 has no entries in the dictionary because it is uniquely distinguished by not being detected by Vector 1. If the dictionary contained any other faults that were not detected by the first vector, fault 8 would require more information in order to be uniquely identified.

For sequential circuits with partially specified output responses, I treat unknown values as though they were distinguishable from logic 0 and logic 1. In other words, a fault producing the signature 0X0 will not be in the same equivalence class as a fault producing the signature 000. Since the signature 0X0 may not actually be distinguishable from the signature 000, fault simulation may stop before a fault has truly been uniquely distinguished. Boppana and Fuchs report that the diagnostic expectation of the resulting dictionary is not substantially affected by this assumption [5].

| Fault | Error set | | |
|---|---|---|---|
| # | 1 | 2 | 3 |
| 1 | 1 | | 2 |
| 2 | | 1 | |
| 3 | | 1 3 | |
| 4 | | 3 | 1 |
| 5 | | | 1 3 |
| 6 | | | 1 3 |
| 7 | 1 | | |
| 8 | | | |

Table 3.9: Further compaction of the dictionary organization is achieved by dropping fully distinguished faults

## 3.5 Circuit and test set characterization

In the following section I compare the sizes of some of the dictionary organizations discussed in Section 3.1 by giving the ratio of the size of the discussed dictionary to the size of the full response dictionary. The size of a full dictionary is computed by multiplying the number of faults by the number of tests by the number of primary circuit outputs. (Since I assume the sequential circuits are not directly resetable, I multiply their full dictionary size by two because there are two bits needed to encode the three possible output values: 1, 0, and unknown). Table 3.10 gives an overview of the ISCAS-85 and ISCAS-89 benchmark circuits [7, 6] and the test sets used. It lists the total number of single stuck-at faults for each circuit, the number of outputs for each circuit, the number of vectors in the test sets for the circuits, the size of the full response dictionary in megabytes, and the diagnostic expectation of a full response dictionary.

The test set for the combinational benchmarks was generated by the DTPG system discussed in Chapter 2. I have experimented with the DTPG test set, a complete single stuck-at test set, a random test set, and a test set from the HITEC test pattern generation system [20]. The test set I used had the worst compression results for the error set

| Circuit | faults | tests | outputs | Full | DE | Circuit | faults | tests | outputs | Full | DE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C432 | 524 | 71 | 7 | .03 | 1.1 | S298 | 308 | 259 | 6 | .11 | 2.5 |
| C499 | 758 | 56 | 32 | .19 | 1.0 | S344 | 327 | 108 | 11 | .09 | 1.9 |
| C880 | 942 | 71 | 26 | .20 | 1.1 | S641 | 465 | 211 | 24 | .56 | 1.5 |
| C1355 | 1574 | 88 | 32 | .53 | 1.9 | S713 | 581 | 175 | 23 | .55 | 2.3 |
| C1908 | 1879 | 146 | 25 | .82 | 1.3 | S820 | 861 | 968 | 19 | 3.85 | 1.6 |
| C2670 | 2747 | 175 | 140 | 8.02 | 1.4 | S832 | 861 | 967 | 19 | 3.77 | 1.6 |
| C3540 | 3428 | 205 | 22 | 1.84 | 1.3 | S1238 | 1372 | 478 | 14 | 2.19 | 1.4 |
| C5315 | 5350 | 172 | 123 | 13.49 | 1.2 | S1423 | 1516 | 88 | 5 | .16 | 9.0 |
| C6288 | 7744 | 57 | 32 | 1.68 | 1.2 | S5378 | 4603 | 900 | 49 | 48.40 | 2.4 |
| C7552 | 7550 | 304 | 108 | 29.55 | 1.2 | S35932 | 39094 | 381 | 320 | 1136.38 | 2.1 |

Table 3.10: Circuit and test set characterization: the number of faults, tests, and primary outputs for each circuit followed by the size of the full dictionary in megabytes and the diagnostic expectation of the test set.

dictionaries. The test set for the sequential benchmarks was generated using the HITEC test pattern generator. (I present results only for sequential circuits with test sets that provide low diagnostic expectation.)

## 3.6 Dictionary results

Tables 3.11 and 3.12 report on the construction of lossless error set dictionaries and error set drop-when-distinguished dictionaries for the twenty listed ISCAS circuits. The first column of each table lists the number of error sets for each circuit. As suggested earlier, the number of error sets is small compared to the maximum possible number of error sets (even the S35932 uses less than 3% of its possible error sets). Next, the tables list the time in seconds to produce the dictionaries. Unlike the Compact method, the new compression technique only requires every fault to be simulated against every vector once (and even fewer simulations are required for the drop-when-distinguished dictionary) [21, 26]. The times needed to generate the new dictionaries are all acceptably small.

Table 3.13 compares the Compact method of Pomeranz and Reddy (labeled Compact)

| Circuit | error sets | time (secs) |
|---|---|---|
| C432 | 127 | 1 |
| C499 | 560 | 3 |
| C880 | 456 | 4 |
| C1355 | 560 | 9 |
| C1908 | 1093 | 20 |
| C2670 | 1111 | 45 |
| C3540 | 2343 | 54 |
| C5315 | 5957 | 122 |
| C6288 | 1970 | 88 |
| C7552 | 4259 | 256 |

| Circuit | error sets | time (secs) |
|---|---|---|
| S298 | 43 | 7 |
| S344 | 439 | 4 |
| S641 | 630 | 8 |
| S713 | 506 | 8 |
| S820 | 371 | 50 |
| S832 | 380 | 50 |
| S1238 | 561 | 30 |
| S1423 | 34 | 16 |
| S5378 | 3694 | 797 |
| S35932 | 337482 | 6940 |

Table 3.11: Lossless dictionary generation results

| Circuit | error sets | time (secs) |
|---|---|---|
| C432 | 90 | 1 |
| C499 | 119 | 1 |
| C880 | 137 | 1 |
| C1355 | 128 | 7 |
| C1908 | 402 | 10 |
| C2670 | 330 | 18 |
| C3540 | 518 | 25 |
| C5315 | 1032 | 35 |
| C6288 | 293 | 28 |
| C7552 | 1066 | 94 |

| Circuit | error sets | time (secs) |
|---|---|---|
| S298 | 32 | 5 |
| S344 | 262 | 2 |
| S641 | 195 | 4 |
| S713 | 152 | 5 |
| S820 | 183 | 32 |
| S832 | 181 | 33 |
| S1238 | 252 | 19 |
| S1423 | 23 | 14 |
| S5378 | 1192 | 500 |
| S35932 | 251532 | 5504 |

Table 3.12: Drop-when-distinguished dictionary generation results

and the $DC2$ diagnostic tree-based scheme of Bopanna and Fuchs (labeled $DC2$) with the lossless error set dictionary (labeled ES) and the drop-when-distinguished error set dictionary (labeled DWD-ES). For each dictionary organization, the table shows the percentage ratio between the dictionary being studied and the full dictionary for the same test set. The table does not report results for the sequential circuits for the Compact dictionary scheme because they were not available. Overall the drop-when-distinguished error set dictionaries are the smallest.

The sizes of the error set-based dictionaries are given based on these decisions: For each fault, the dictionary stores indexes into a vector list and an error set table. The number of bits required to store these indexes varies with the sizes of the tables, but for simplicity, I used 16-bit data-items for all results reported here. The use of 16-bit data-items leaves

| Circuit | Compact FF % | $DC2$ FF % | ES FF % | DWD-ES FF % |
|---|---|---|---|---|
| C432 | 31.2 | 23.6 | 49.2 | 17.2 |
| C499 | 5.0 | 7.7 | 24.3 | 6.4 |
| C880 | 15.6 | 7.3 | 14.6 | 4.8 |
| C1355 | 4.0 | 5.1 | 17.9 | 13.5 |
| C1908 | 7.0 | 4.9 | 16.8 | 6.7 |
| C2670 | 2.3 | 1.1 | 2.0 | 0.6 |
| C3540 | 18.2 | 5.5 | 11.9 | 4.0 |
| C5315 | 7.5 | 1.5 | 2.9 | 0.6 |
| C6288 | 45.5 | 8.9 | 22.6 | 9.0 |
| C7552 | 6.5 | 1.3 | 2.5 | 0.8 |

| Circuit | $DC2$ FF % | ES FF % | DWD-ES FF % |
|---|---|---|---|
| S298 | 9.2 | 27.4 | 20.8 |
| S344 | 6.5 | 37.8 | 22.9 |
| S641 | 2.9 | 6.9 | 2.8 |
| S713 | 2.9 | 6.9 | 3.8 |
| S820 | 2.8 | 4.0 | 1.9 |
| S832 | 2.8 | 4.1 | 1.9 |
| S1238 | 4.1 | 3.4 | 1.3 |
| S1423 | 11.5 | 5.5 | 4.7 |
| S5378 | 1.2 | 1.9 | 0.9 |
| S35932 | 0.4 | 1.9 | 1.4 |

Table 3.13: Dictionary size as a percentage of the size of the corresponding full dictionary for combinational and sequential circuits is given for Compact, the $DC2$ diagnostic tree, error set-based, and drop-when-distinguished error set-based dictionaries.

plenty of room for expansion: much larger circuits could be processed without needing to use larger data-items.

The size of an error set dictionary is related to the number of detections the dictionary records. Figure 3.3 graphs two quantities for the ISCAS-85 dictionaries: the number of data items that comprise each dictionary and the number of fault detections each dictionary stores. By representing the two sets of data as continuous rather than discrete functions, the relationship between the quantities is emphasized. The C2670 demonstrates the power of the elided sequence optimization discussed in Section 3.3. There are actually fewer data items than fault detections. Table 3.13 shows that the sizes of the competing dictionary schemes fluctuate in the same way. That is, certain circuits inherently lend themselves to small dictionaries.

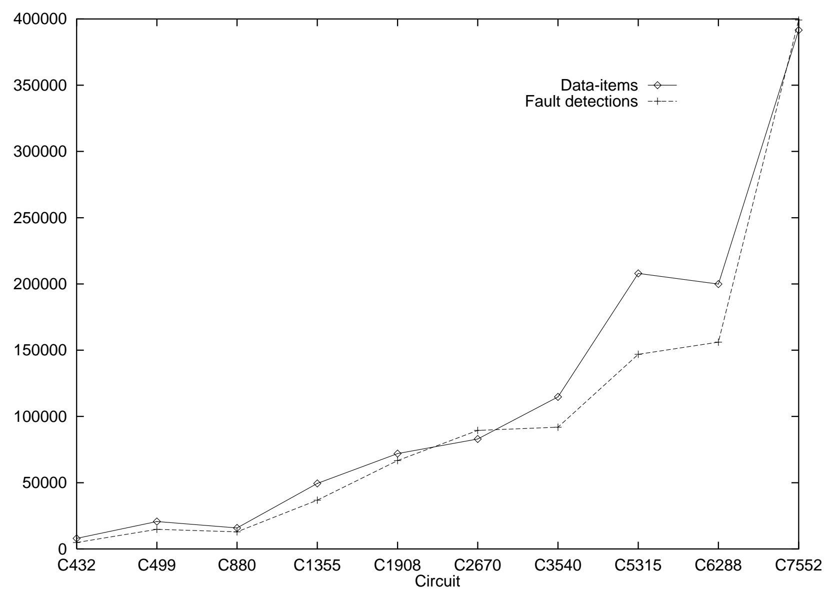


Figure 3.3: The number of data items in an error set dictionary tracks the number of fault detections the dictionary stores.

# 4. Diagnosis of faults not modeled by the dictionary

There are arguments for diagnosing realistic faults directly by making a dictionary comprised of realistic faults [4], but there are also several arguments against creating such dictionaries. The number of realistic faults in a circuit is significantly larger than the number of single stuck-at faults for a circuit, and the cost of simulating an individual realistic fault is usually much greater than the cost of simulating a stuck-at fault [10]. Furthermore, accurate modeling of realistic faults is critical to being able to diagnose them directly. A simulator that does a good job of fault grading may not have the accuracy required to produce a useful realistic fault dictionary. Finally, it is unlikely that a fault dictionary could be made to contain a behavior for every type of realistic fault under all conditions. It is important to consider the coverage of unmodeled faults when examining fault coverage [28], and it is even more important to address the performance of a fault dictionary with regard to faulty behaviors it does not contain.

It is popular to investigate dictionary compaction methods that reduce the amount of information in the dictionary but do not affect the sizes of the equivalence classes contained in the dictionary. How much does eliminating this supposedly redundant information reduce the ability of the dictionary to diagnose unmodeled faults?

In order to answer this question, I performed the following experiment. First, I created four stuck-at fault dictionaries for the MCNC layouts of the ISCAS-85 benchmarks:

1. A full dictionary.

2. A drop-when-distinguished dictionary.

3. A drop-on-10 dictionary.

4. A drop-on-5 dictionary.

Figure 4.1 shows the diagnostic expectation of these dictionaries for the MCNC versions of the ISCAS-85 circuits. Note that the diagnostic expectation of the full dictionary and the drop-when-distinguished dictionary are identical, because the drop-when-distinguished dictionary does not lose any diagnostic resolution for the faults in the dictionary. As expected, the drop-on-10 dictionary has greater diagnostic expectation than the full dictionary, and the drop-on-5 dictionary is worse than any of the other dictionaries. The drop-on-5 and drop-on-10 dictionaries roughly track the size of the drop-when-distinguished dictionary, though the drop-on-5 dictionary is slightly smaller for all circuits.

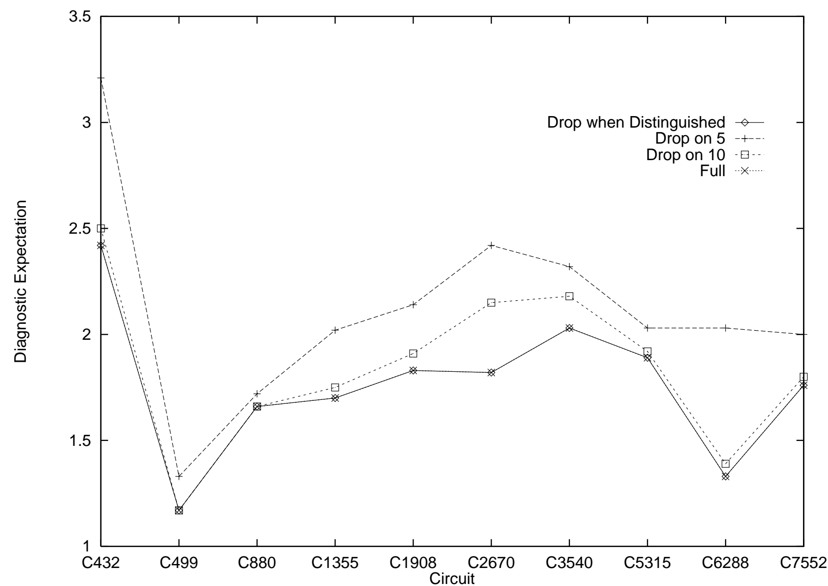


Figure 4.1: Diagnostic expectation for four dictionaries. (The diagnostic expectat ion for the drop-when-distinguished dictionary is identical to the diagnostic ex pectation for the full dictionary.)

The Nemesis bridging fault simulator produced the observed faulty behaviors used to evaluate the dictionaries [9]. The behavioral model used by Nemesis for bridging faults

is two-component simulation, in which the gates driving the bridged nodes are SPICE simulated to determine the bridge voltage, which is then compared against the SPICE-computed logic thresholds of downstream gates in order to model the effect of the Byzantine Generals Problem [2, 3]. The simulator also extensively models feedback bridging faults. If a feedback bridging fault evidences the potential to oscillate or hold state, the simulator forces the bridge voltage to be the fault free value on the rear bridged node, thereby disallowing oscillation.

Because the faulty responses being diagnosed are not found in the dictionaries, I need a procedure for determining which wires are involved in the targeted fault. That is, if wire A and wire B are bridged together, the procedure should identify that the fault involves wire A, wire B, or both. This experiment was modeled after the Teradyne fault diagnosis system [22], which penalizes candidate faults for each predicted failure that did not actually occur and for each failure that does occur without being predicted by the candidate. This procedure produces a ranked list of faults. Given this ranking, if any of the four possible stuck-at faults associated with the two bridged nodes appears among the ten highest ranked faults, the diagnosis is considered a success, otherwise it is considered a failure.

I did not use the technique of Millman, McCluskey, and Acken for diagnosing bridging faults with stuck-at fault dictionaries [19] because I wanted to design an experiment that was blind to the type of defect being diagnosed. For instance, if the faults being diagnosed were breaks rather than bridges, the experimental design would not change.

I attempted to diagnose signatures for the 10% of realistic bridging faults determined to be most likely to occur by the inductive fault analysis tool Carafe [15]. The results of the

experiment are shown in Figure 4.2. As expected, the full dictionary always has the fewest diagnostic failures, followed by the drop-on-10 dictionary and the drop-on-5 dictionary. Surprisingly, the drop-when-distinguished dictionary has the most failures for all circuits except the C432. Figure 4.1 makes it appear that the full dictionary and the drop-when-distinguished dictionary have the same diagnostic ability—and that is true when the realm of diagnosis is restricted to single stuck-at faults—but the drop-when-distinguished dictionary is much less effective when applied to faults that it does not model. I have shown that it does a very poor job of diagnosing a common type of defect that actually appears in CMOS circuits.

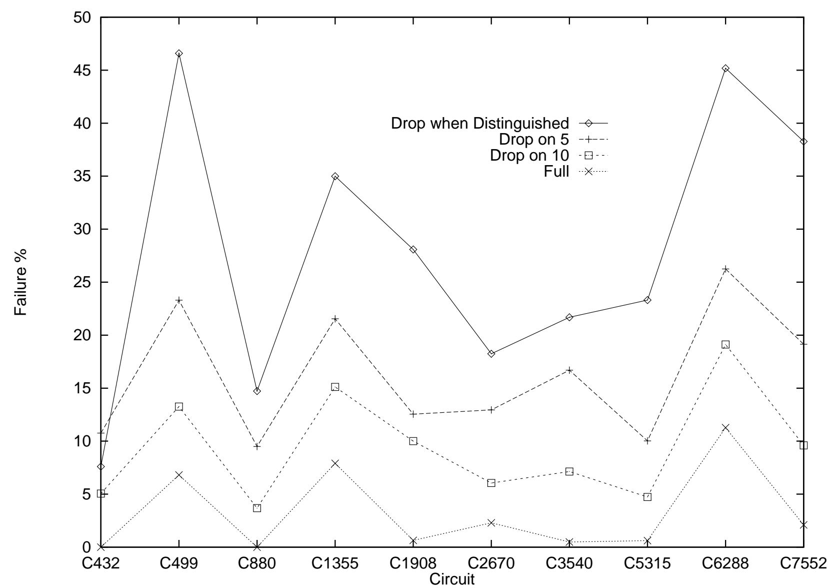


Figure 4.2: Diagnostic failure for four dictionaries.

# 5. Conclusions

Diagnostic test pattern generation allows us to build dictionaries with the best possible diagnostic expectation. I have provided the minimum size of the fault equivalence classes for the ISCAS-85 benchmark circuits. Some of the equivalence classes suggest that new methods of fault collapsing based on local topology may be possible.

There are trade-offs between dictionary size, dictionary computation time, and dictionary diagnostic content. The concept of an error set is a new idea that allows us to efficiently produce compressed dictionaries with maximum diagnostic content. The error set-based approach can be combined with other compaction techniques, which do not sacrifice conventionally-measured diagnostic expectation but do sacrifice diagnostic information in order to decrease dictionary size. For the largest benchmark circuits, the resulting error set-based dictionaries are generally smaller than any available alternative.

I claim that, regardless of the technique, removing information from a dictionary will cause it to do a poorer job of diagnosing faults that were not used to construct the dictionary. I have presented an experiment to support this assertion. It is clear that the standard metrics of diagnostic quality on the faults modeled in the dictionary are inadequate for assessing the dictionary's ability to diagnose unmodeled faults. The testing community needs a better way to evaluate the tradeoffs involved in dictionary construction so that diagnosis researchers do not become too far removed from the need for precise and accurate diagnosis of real defects.

# References


[1] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital Systems Testing and Testable Design*. Computer Science Press, 1990.

[2] J. M. Acken and S. D. Millman. Accurate modeling and simulation of bridging faults. *Proceedings of the Custom Integrated Circuits Confernce*, pages 17.4.1–17.4.4, 1991.

[3] J. M. Acken and S. D. Millman. Fault model evolution for diagnosis: Accuracy vs precision. *Proceedings of the Custom Integrated Circuits Confernce*, 1992.

[4] R. C. Aitken and P. C. Maxwell. Better models or better algorithms? on techniques to improve fault diagnosis. *Hewlett-Packard Journal*, February 1995.

[5] V. Boppana and W. K. Fuchs. Fault dictionary compression by output sequence removal. In *Proceedings of International Conference on Computer-Aided Design*, pages 576–579. IEEE, 1994.

[6] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential circuit benchmarks. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, 1989.

[7] F. Brglez and H. Fujiwara. A neutral netlist of 10 combinational benchmark circuits and a target translator in fortran. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, 1985.

[8] P. Camurati, D. Medina, P. Prinetto, and M. Sonza Reorda. A diagnostic test pattern generation algorithm. In *Proceedings of International Test Conference*, pages 52–58. IEEE, 1990.

[9] B. Chess and T. Larrabee. Bridge fault simulation strategies for CMOS integrated circuits. In *Proceedings of Design Automation Conference*, pages 458–462, 1993.

[10] B. Chess, T. Larrabee, and C. Roth. On evaluating competing bridge fault models for CMOS ICs. In *Proceedings of the 1994 VLSI Test Symposium*, pages 446–451. IEEE, 1994.

[11] D. Feltham and W. Maly. Physically realistic fault models for analog CMOS neural networks. *IEEE Journal of Solid-State Circuits*, 26(9):1223–1229, September 1991.

[12] F. J. Ferguson, M. Taylor, and T. Larrabee. Testing for parametric faults in static CMOS circuits. In *Proceedings of International Test Conference*, pages 436–443. IEEE, 1990.

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

[14] T. Gruning, H. Koopmeiners, and U. Mahlstedt. DIATEST: A fast diagnostic test pattern generator for combinational circuits. In *Proceedings of International Conference on Computer-Aided Design*, pages 194–197. IEEE, 1991.

[15] A. Jee and F. J. Ferguson. Carafe: An inductive fault analysis tool for CMOS VLSI circuits. In *Proceedings of the IEEE VLSI Test Symposium*, pages 92–98, 1993.

[16] K. Kubiak, S. Parkes, W. K. Fuchs, and R. Saleh. Exacts evaluation of diagnostic test resolution. In *Proceedings of Design Automation Conference*, pages 347–352. IEEE, 1992.

[17] T. Larrabee. Test pattern generation using boolean satisfiability. *IEEE Transactions on Computer-Aided Design*, pages 4–15, January 1992.

[18] U. Mahlstedt, T. Gruning, W. Daehn, and C. Ozcan. CONTEST: A fast atpg tool for large combinational circuits. In *Proceedings of International Conference on Computer Design*, pages 222–225. IEEE, 1990.

[19] S. D. Millman, E. J. McCluskey, and J. M. Acken. Diagnosising CMOS bridging faults with stuck-at fault dictionaries. In *Proceedings of International Test Conference*, pages 860–870. IEEE, 1990.

[20] T. Niermann and J. Patel. HITEC: A test generation package for sequential circuits. In *PrEDAC*, pages 214–218, 1991.

[21] I. Pomeranz and S. M. Reddy. On the generation of small dictionaries for fault location. In *Proceedings of International Conference on Computer-Aided Design*, pages 272–279. IEEE, 1992.

[22] J. Richman and K. R. Bowden. The modern fault dictionary. In *Proceedings of International Test Conference*, pages 696–702. IEEE, 1985.

[23] J. P. Roth. Diagnosis of automata failures: A calculus and a method. *IBM Journal of Research and Development*, 10:278–291, 1966.

[24] E. Rudnick, W. K. Fuchs, and J. Patel. Diagnositic fault simulation of sequential circuits. In *Proceedings of International Test Conference*, pages 178–186. IEEE, 1992.

[25] P. Ryan, S. Rawat, and W. K. Fuchs. Two-stage fault location. In *Proceedings of International Test Conference*, pages 963–968. IEEE, 1991.

[26] P. G. Ryan, W. K. Fuchs, and I. Pomeranz. Fault dictionary compression and equivalence class computation for sequential circuits. In *Proceedings of International Conference on Computer-Aided Design*, pages 508–511, 1993.

[27] M. H. Schulz, E. Trischler, and T.M. Sarfert. SOCRATES: a highly efficient ATPG system. *IEEE Transactions on Computer-Aided Design*, CAD-7(1):126–137, January 1988.

[28] L. W. Wang, M. R. Mercer, T. W. Williams, and S. W. Kao. On the decline of testing efficiency as fault coverage approaches 100%. In *Proceedings of the 1995 VLSI Test Symposium*, pages 74–83. IEEE, 1995.