A Hierarchical Method of Fault Diagnosis with Built–In Self–Test Applications

Robert Campbell Aitken

B.Sc. (Hon), University of Victoria, 1985
M.Sc., University of Victoria, 1986

Department of Electrical Engineering
McGill University

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

April 1990

Abstract

The problem of fault diagnosis in digital circuits is composed of two sub-problems: Fault detection (identifying that a fault is present) and fault location (identifying the failure responsible for faulty behaviour) Tests using randomly or pseudo-randomly selected input stimuli have been suggested both as a means of reducing the costs associated with generating deterministic tests and as a means by which circuits could test themselves (built-in self-test). The majority of research into random testing has dealt only with its fault detection properties. Fault location has typically been treated as a dictionary search problem, or ignored altogether.

This dissertation proposes a new approach to fault location in randomly or pseudo-randomly tested digital circuits. Faults will be isolated by a hierarchical sequence of steps, each of which identifies a particular property of the fault which has occurred. The method uses available circuit information to aid in location. The advantages of this method include greatly reduced dictionary generation costs, the ability to locate a fault without searching an entire dictionary, and the ability to partially characterize unmodelled faults In addition, a hierarchical approach allows dictionaries to be constructed in a demand-driven fashion, avoiding unnecessary work for faults which do not occur in practice, and reducing the costs of changes in the test set.

Sample applications of the proposed location technique are given, for use in both random compact testing and built-in self-test Experimental results are presented to demonstrate the performance of the method.

Résumé

Le diagnostique de défauts dans les circuits digitaux se divise en deux sous problèmes la détection des défauts et leur identification. Des méthodes qui utilisent les entrées aléatoires ont été proposés comme façon de réduire les coûts associés à la génération des patrons de test et pour faciliter le test intégré. La plupart des recherches reliées aux tests aléatoires ont seulement considéré la détection des défauts L'identification des défauts a été considérée comme un problème de recherche en dictionnaire ou tout simplement ignorée.

Ce document propose une nouvelle approche à la question de l'identification des défauts dans les circuits digitaux vérifiés à l'aide de patrons de test aléatoires. Les défauts sont isolés par une séquence hiérarchique d'étapes où chacune identifie un attribut particulier du défaut qui est présent dans le circuit. La méthode utilise les caractéristiques connues du circuit pour faciliter l'identification des défauts. Les avantages de cette méthode sont entre autre une grande réduction des coûts de génération de dictionnaire, la possibilité d'isoler un défaut sans la recherche complète du dictionnaire et l'abilité de caractériser partiellement les défauts qui n'ont pas été modelé De plus, une approche hiérarchique permet aux dictionnaires d'être construits sur demande évitant ainsi des efforts inutiles et réduisant les coûts associés aux modifications du test

Ce mémoire contient quelques exemples des applications envisagées ainsi qu'une évaluation expérimentale de la méthode proposée.

Acknowledgements

I would like to acknowledge my supervisor, Vinod K. Agarwal, for his assistance and guidance throughout my doctoral studies. His enthusiasm and vision have been greatly appreciated, as has his insistence on and support for publication of results. The latter has enriched this work and broadened my experience as a researcher considerably

I would also like to acknowledge Prof. Janusz Rajski for his suggestions relating to the organization of this dissertation. In addition, as members of my supervisory committee, both he and Prof. Hossam ElGindy contributed worthwhile suggestions during the earlier stages of my research work.

I am indebted to my friends in the VLSI Design Lab, present and past members, for their assistance — notably to Fidel Muradali, for proofreading the various drafts of this dissertation, and also to Henry Cox, André Ivanov, Fadi Maamari, Ashish Pancholy, Dhirendran Xavier, Yervant Zorian, and to Francis Larochelle for his assistance in writing the French version of the abstract.

In addition, I would like to express my thanks to Jon Muzio and Micaela Serra, of the University of Victoria, for introducing me to the testing field.

For financial support, I would like to acknowledge the Natural Sciences and Engineering Research Council of Canada, and the Centre de recherche informatique de Montréal.

Finally, on a more personal level, I am grateful to my wife Denise Aitken for her support throughout the course of my studies, including delaying some of her own goals in order to help me realize mine

Table of Contents

List of Figures

List of Tables

Claim of Originality

The author claims originality for the following contributions of the dissertation·

- Chapters 1 through 4 are largely review, although they are original in the sense of providing a unified background to the field of fault diagnosis in randomly-tested circuits employing data compaction (signature-tested circuits).
- Chapter 3 includes a description of the asymmetric error model which was proposed for built-in self-test by the author.
- Chapter 4 includes a novel set of criteria to be used when evaluating fault location methods for signature-tested circuits.
- Chapter 5 develops a new model for fault coverage curves, which permits the estimation of the relative effort required in fault simulation when faults are dropped after detection or simulated over an entire test set
- Chapter 5 proposes a hierarchical methodology for fault location in signature-tested circuits. The methodology involves a succession of reduction steps on an initial fault set and represents a marked departure from previous efforts in this area.
- Chapter 5 also suggests new signatures for random testing which contain useful fault location information, and develops a new technique for evaluating the location capability of these in the context of demand-driven fault diagnosis.
- Chapter 6 suggests circuit structures to implement these signatures. These adaptations are novel, with the exception of the partial sequence matching method suggested for reducing hardware overhead, which has been applied previously to the output data modification technique of built-in self-test. However, a new algorithm for producing a partial matching is developed
- Chapter 7 gives a detailed description of application techniques for hierarchical fault diagnosis, including expressions for expected values, variance, and aliasing

probability of signatures. Together, these new methods are suitable for a wide variety of signature-tested circuits.

- Chapter 8 contains a detailed cost comparison between the proposed hierarchical fault location method and two other methods previously proposed for fault location in signature-tested circuits. To the author's knowledge, such cost evaluations have never been reported for either of the previously-developed methods.
- Chapter 9 provides extensive experimental verification of the performance of the hierarchical fault location technique. Since the techniques are new, the evaluation is also novel.
- Appendix A gives exact and asymptotic expressions for the aliasing probability of the weight counting BIST technique for both the independent and asymmetric error models. Standard combinatorial techniques were used to derive these, but the expressions themselves are new.

Chapter 1 Introduction

This dissertation develops a new method for performing fault diagnosis in combinational digital circuits tested with random or pseudo-random input vectors. Fault detection is accomplished using signatures from output data compaction, and fault location is performed hierarchically using each signature to reduce the fault set. The major contributions of this method include the development of signatures which contain useful circuit information, the ability to use this information for diagnosis, and the potential to implement the diagnosis hardware on the chip or module being tested. These properties allow the proposed method to overcome what has has previously been a major drawback to the use of data compaction techniques in fault diagnosis — the difficulty of performing fault location. The vocabulary of fault diagnosis as well as a brief introduction to the field are given in the following section.

1.1 Definitions

A typical digital circuit contains inputs, internal logic, and outputs. Digital signals, represented by a set of binary 1s and 0s called an *input vector* or *input pattern*, are applied at the inputs, and the internal logic's response, correspondingly referred to as an *output vector* or *output pattern*, is read at the outputs. If the output function of the circuit is determined solely by its present input vector, the circuit is called *combinational*. If, on the other hand, the output function is determined by the sequence of input vectors applied, then the circuit contains memory and is termed *sequential*.

Manufacturing digital circuits is a complex process, and, as a result, some of the circuits produced are faulty. A *fault* is defined as the cause of incorrect circuit behaviour. *Fault diagnosis* for digital circuits can be defined as answering the question "Which, if any, faults are present in this circuit?" Fault diagnosis encompasses the testing of digital circuits, and is in fact composed of two subproblems: *fault detection*, "Is there any fault present in this circuit?" and *fault location*, "Which of the many possible faults is (are) responsible for the incorrect behaviour of this circuit?"

Fault diagnosis is needed during the manufacturing of digital circuits — whether the systems (computers and other machines) themselves, or the boards, cards and modules of which they are comprised, or the integrated circuits (ICs) or "chips" which in turn make up the boards. This distinguishes fault diagnosis from the problem of *design verification*, which answers the question "Does this design perform the function it is intended to perform?" Testing assumes that design verification has already been performed, so that if the circuit contains no faults it will implement the correct function.

The goal of fault diagnosis, naturally, is to find faulty circuits as early as possible in the manufacturing process. The cost of a fault increases by approximately an order of magnitude at each step of the process from silicon wafer to chip to package to board to system to shipped product [Wil83] The complexity of modern circuits has increased the difficulty of fault diagnosis to such an extent that testing now accounts for about a third of the cost of an IC [Bha89].

The faults to be diagnosed may be *logical*, which cause changes in a faulty circuit output function known as *errors*, or *parametric*, which cause some parameter of circuit operation, such as voltage level, timing, or capacitance, to degrade beyond the specified minimum level for the design. Faults may be *permanent*, *transient* (occurring only once), or *intermittent* (recurring). Faults may be either combinational or sequential in nature, independent of the circuit under test. Faults can result from either physical defects or electrical failures within the manufactured circuits. A circuit without any such failures is said to be *fault-free*.

The failures, or defects, are caused by a variety of physical phenomena: Many of today's chips contain elements which are smaller than 1 μm — one millionth of a meter — across. These can have their properties altered by minor imperfections in the silicon, be destroyed by microscopic dust fragments, or be obliterated by tiny misalignments in the "masks" used to create them. Additional defects, such as extra or missing connections, can occur when the chips are packaged, and still more are possible when the chips are installed on circuit boards.* Not all defects result in faults, only those which cause some sort of erroneous circuit behaviour.† Fault diagnosis attempts to discover such behaviour and pinpoint its cause

* For more information on IC defects, see [Rav81] [McC87]

† Certain faults, called *redundant faults* cause no observable change in circuit output on their own, but may cause errors if they occur together with other faults [Fri67].

This dissertation is primarily concerned with the diagnosis of permanent logical faults in combinational circuits. Where the results obtained may be applied to other forms of diagnosis, the fact is noted, but the primary focus remains.

1.2 Overview of Problem

As integrated circuits have increased in complexity, testing has become a significant production expense, and now represents on the order of one third of the total cost of an IC. The expense arises from both the hardware required to perform the tests, and the effort needed to develop them. The primary focus for fault diagnosis strategies has been deterministic test pattern generation (DTPG) techniques, where test stimuii are designed explicitly to detect (and possibly locate) modelled faults

In an effort to avoid the costs of DTPG, random test patterns have been proposed as an alternate testing method. A random or pseudo-random source generates input stimuli for the circuit under test, and faults are detected implicitly A sufficiently long random test will contain test patterns which detect all faults detected by its deterministically generated equivalent. Random test sets are often considerably longer than deterministic test sets.

Data compaction techniques are often used in random testing in order to reduce the amount of data resulting from the large number of input patterns. Data compaction and random test stimuli can greatly simplify test generation and hence reduce test development costs. In addition, the hardware required by the methods is simpler than that needed for deterministic testing, so test application costs may be reduced, and test hardware may be included as part of the circuit under test. This last approach is known as built-in self-test (BIST). When the hardware is external, the technique is referred to as random compact testing (RCT).

One failing of both BIST and RCT has been that while fault detection is straightforward, fault location in the presence of data compaction has been difficult and expensive Fault location is required for repairable or fault tolerant systems, and also to evaluate and improve the manufacturing process for conventional circuits Several techniques have been suggested previously, but none of these is able to satisfy all the requirements of such methods. This dissertation proposes data compaction techniques and a fault location method which meets these criteria, and in addition can reduce location effort significantly over the previous approaches

The essential feature of the proposed method is its hierarchical nature: the set of possible faults is reduced over a sequence of steps to include only those faults consistent with the various observed circuit behaviours. The data compaction process ensure that useful information about faults present in the circuit is readily available — a result which has not been possible with conventional approaches.

1.3 Outline of Dissertation

The remainder of this dissertation describes the hierarchical fault diagnosis method in detail, and is organized as follows:

Chapter 2 provides a brief history of fault diagnosis, reviewing the literature on the subject as it has evolved and giving a general background to the field.

Chapter 3 reviews built-in self-test of circuits, and includes sections on error modelling and data compaction techniques, which apply to both self-testing and randomly-tested circuits. These techniques provide a basis for the signatures proposed in this dissertation.

Chapter 4 reviews the methods which have been proposed previously for locating faults in a signature testing environment, which includes random input vectors and output data compaction. The chapter examines their relative advantages as well as their shortcomings, and concludes with set of criteria which must be met by any fault location method employing data compaction.

Chapter 5 introduces the hierarchical fault location method which is the focus of the dissertation. It also provides a formal methodology for analyzing and comparing fault location techniques. This is used to show the benefits of the hierarchical approach with a variety of possible signatures. In addition, a model for calculating the fault and compactor simulation effort required by dictionary construction is developed.

Chapter 6 discusses the physical structures necessary to observe the circuit parameters mentioned in chapter 5 as potential signatures Tradeoffs between potential diagnostic resolution and the amount of hardware overhead are investigated.

Chapter 7 uses the signatures examined in chapter 5 and the structures of chapter 6 to develop sample applications of the hierarchical fault location method The techniques apply to both random compact testing and built-in self-test and are examined in detail. Means of calculating expected values for each signature given an error model and fault detection probabilities are shown, and the uses of each of the sample applications are discussed

Chapter 8 compares the costs (including start-up and run-time costs in terms of time, space and hardware complexity) of the hierarchical fault location method with those of the techniques reviewed in chapter 4. The superiority of the hierarchical approach i demonstrated for a variety of cost criteria and potential applications

Chapter 9 outlines some experiments conducted to validate the performance projections of earlier chapters. The circuits used belong to a standard set of combinational benchmarks.

Finally, chapter 10 concludes the dissertation and outlines open problems in the area.

Chapter 2 A Brief History of Fault Diagnosis

Most literature on fault diagnosis is concerned with permanent logical faults. Parametric testing is important and has not been entirely ignored (see, for example, [Zas85] [Mal88] [Nig90]). Similarly, intermittent faults have been considered (e.g. [Bal69] [Yen69] [Kam75] [Sav80b]), but nonetheless the trend remains. This chapter reviews the progress of fault diagnosis in digital circuits throughout the last thirty years, showing the changes in methods and in the perception of the issues involved as circuit complexity has increased from logic gates constructed from vacuum tubes to today's ultra-large scale integration and beyond.

2.1 Early History

Initially, fault detection and fault location were essentially the same process. A circuit was given a test, which consisted of applying an input vector (for combinational circuits) or sequence of input vectors (for sequential circuits), then comparing the output vector(s) with some expected results. Each test would check for a particular possible physical failure [Tsi62] [Cha65]. However, many input vectors are able to test for several faults, even in simple circuits. So, lists were created, showing for every possible input vector the corresponding effect that each fault would have on the circuit output. These lists are called *fault dictionaries*. (The earliest reference appears to be [Tsi62], which refers to them as "maintenance dictionaries") These dictionaries were originally issued as books, and the individuals performing the tests would scan through them to diagnose any faults in the system they were observing. Even the earliest dictionaries were large — [Dow64] describes that of [Tsi62] as being 1200 pages long

As circuits became more complex, such dictionaries became impractical. A record of all possible faulty output values for all possible inputs is almost always an excessive and unnecessary amount of information for the diagnosis process. Throughout the 1960s, work continued on reducing, or better, minimizing, the required dictionary size ([Ses62] [Ses65] [Cha65] [Arm66] [Kau68]). Fault diagnosis was frequently treated as an extension of the well-known boolean function minimization problem [McC56]. Such treatment required that the full dictionary be known, even if it was not physically produced. Its construction was assumed to be straightforward. as evidenced by this offhand remark in [Kau68]: "One has no difficulty in imagining that an analysis of [the circuit] has been conducted in order to determine the effect that each of various hypothetical faults has on its output." Today this is beyond imagination, even for combinational circuits. Circuits with 75 inputs are not uncommon. Such a circuit has a total of 2^{75} possible input combinations. If one billion of these combinations could be analyzed every second, it would take more than a million years to complete the dictionary. Each additional input would double both the generation time and the storage required.

2.2 Test Sets

The excessive cost of calculating a complete dictionary led to a search for approaches other than minimization to the problem of generating smaller dictionaries. The smaller dictionary would list faulty responses only for a subset of possible input combinations called the *test set*. The vectors in a test set should both detect each potential fault and permit fault location. When developing a test set, both a circuit and its potential faults may be considered at various hierarchical levels of abstraction.

2.2.1 Hierarchical Levels of Circuit Description

There are two main classes of circuit descriptions: *functional* and *structural*. A functional description of a computer register, for example, could contain a list of the various operations possible on it (load, store, add, etc.), while a structural model could list the transistors which made up the register and the interconnections and signals between them. Various of levels of description are possible for both functional and structural models. Registers may be combined into modules such as memories or decomposed into their constituent elements, called latches or flip-flops. Transistors may be grouped together to form gates performing logic functions such as AND, OR and NOT, or decomposed into the silicon and metal layers which form them. The hierarchical nature of these models is shown in figure 2.1. More information on digital circuit design may be found in [Man79] [Mea80] and [Wes85].

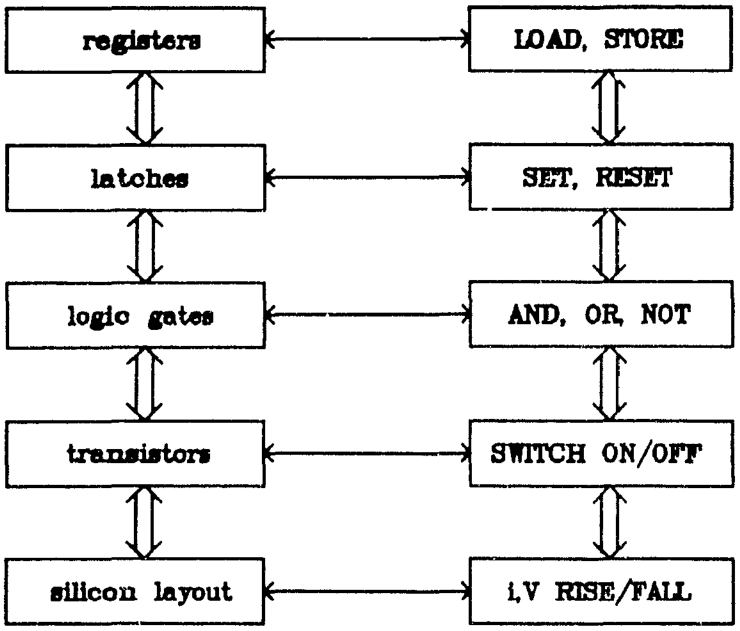


Figure 2.1 Hierarchy of Structural and Functional Circuit Models

2.2.2 Hierarchical Levels of Fault Description

The fault description depends on the circuit description. The idea of reading an incorrect address within memory has no direct meaning at the transistor level, and an electrical short between a transistor input and ground is similarly without direct meaning at the register transfer level, although both may describe the same physical failure. Test sets are designed to detect particular sets of faults which are determined by a *fault model.*

Most test sets are developed at the structural level rather than the functional level [Bha89], since structural tests are believed to be much more thorough than functional tests. The fault model most frequently used is the *gate-level stuck-at model.*

In the gate-level stuck-at fault model, inputs or outputs of logic gates within a circuit may be "stuck-at", i.e. permanently set to, either 0 or 1, which is analogous to being shorted to either ground or power respectively. Table 2.1 shows the behaviour of some common logic gates in the presence of stuck-at faults. The stuck-at model was introduced by Eldred in [Eld59] as a suggested fault model for testing logic gates comprised of vacuum tubes and diodes. While such gates have long since passed into history, the model remains. Various authors have since questioned its applicability, e.g. [Cle71] [Sus73] [Mei74] [Wad78a] [Gal80] [Wil83] [She85] [McC87], and additional models have been proposed, among them bridging faults [Mei74], where lines are shorted to one

f	AB	C	A/0	A/1	B/0	B/1	C/0	C/1
AND	00	0	0	0	0	0	0	1
	01	0	0	0	0	1	0	1
	10	0	0	1	0	0	0	1
	11	1	0	1	0	1	0	1
OR	00	0	0	1	0	1	0	1
	01	1	1	1	0	1	0	1
	10	1	0	1	1	1	0	1
	11	1	1	1	1	1	0	1
XOR	00	0	0	1	0	1	0	1
	01	1	1	0	0	1	0	1
	10	1	0	1	1	0	0	1
	11	0	1	0	1	0	0	1

Table 2.1 Gate Behaviour in the Presence of Stuck-At Faults

another, CMOS stuck-open [Wad78a] and transition faults [She85], based on extra or missing connections at the transistor level, and crosspoint faults [Smi79], which result from connectivity errors in PLAs. Nonetheless, the stuck-at model, and in particular the single stuck-at model, where at most one fault is assumed to occur within a circuit, remain the most popular [Bha89]. The most common justification for the continued use of the model is that complete tests for single stuck-at faults tend to be excellent tests for other types of faults as well [Wil83] [Raj85] [Jac86].

2.2.3 Quality Measurements

With the preponderance of methods available for generating test sets, it is necessary to have some means of comparing them. Besides the obvious criteria of test length and test generation time, the quality of the methods can be compared

2.2.3.1 Fault Coverage

The fault detection capability, or *fault coverage* of a test set is usually defined as

$$\text{Fault coverage} = \frac{\text{Faults detected by test set}}{\text{Total modelled faults}}$$

Confusion over the number of modelled faults (including or not including equivalent faults, redundant faults, dominated faults, etc.) can affect the fault coverage value

[Aga80] [Cox88], but the measure remains useful.* A test set is considered to be *complete* with respect to a fault model if all irredundant faults within the model are detected.

2.2.3.2 Diagnosability

The fault location capability of different test generation methods can also be compared. The size of the fault dictionary, complexity of creating it, and the resolution of the method are all important comparison criteria. Mandelbaum [Man64] has suggested using the entropy function of communication theory [Sha48] for resolution comparisons. This is the function used in this dissertation. For a diagnosis method which potentially divides the faults into N classes, the diagnosability function is given as:

$$R = -\frac{1}{\log h}\sum_{i=1}^{N} g(i)\log(g(i))$$

where

$$h = \text{total number of modelled faults}$$
$$g(i) = \frac{1}{h} \times \text{number of faults in class } i$$

and the indeterminate $0 \log 0$ is defined to be 0.

This function has minimum value 0 when all faults are in the same class, and maximum value 1 when each forms its own class.

2.2.3.3 Relationships with Circuit Yield

The number of faulty circuits released for sale is exponentially affected by test quality, according to Williams and Brown [Wil81], who give the defect level, DL, of a group of manufactured circuits as:

$$DL = 1 - Y^{1-T}$$

where Y is the yield of the manufacturing process and T is the fault coverage of the testing process. High fault coverage is clearly an important property of any test set, particularly when process yield is low For the formula to be accurate, it is important that the figure T be the coverage of actual failures, not necessarily the coverage figure from a particular fault model

* Other fault coverage measures based primarily upon circuit topology and likely failure points have been proposed, e.g [Wad78b], but their technology-dependent nature has kept them from common use

2.3 Aspects of Automated Test Pattern Generation

In order to make the testing process more scientific, methods have been proposed to produce a test set automatically for a given circuit. Two classical techniques for this *automated test pattern generation* (ATPG) were characterized as *test pattern generation* and *fault insertion* by Susskind in [Sus73]. In current terminology, these are *deterministic test pattern generation* and *random test pattern generation* respectively. Despite the fact that most manufactured circuits contain sequential elements, much research has gone into investigating ATPG methods for combinational circuits. There are two reasons for this: First, test generation for combinational circuits is an easier problem, and second, scan techniques (see section 2.6) can convert a sequential circuit to a combinational one for the purpose of testing.

2.3.1 Deterministic Test Pattern Generation

Deterministic test pattern generation (DTPG) does not require a physical implementation of the circuit. Instead, a test set is derived algorithmically based on a circuit description and a fault model. The fault coverage of this test set may be verified analytically or by simulation. The DTPG techniques investigated apply primarily to combinational circuits.

2.3.1.1 DTPG Methods

Deterministic test pattern generation using the gate-level stuck-at model has existed since the early 1960s. The idea of path sensitization was reportedly in existence in 1961 [Arm66]. The D-Algorithm [Rot66] (and its extension to sequential circuits in [Bou71]), boolean difference [Sel68], PODEM [Goe81], FAN [Fuj83], critical path tracing [Abr84], Socrates [Sch88], and CAMP [Raj90] are just some of the test pattern generation strategies developed over the years.

2.3.1.2 Complexity of Test Pattern Generation

The development of new test pattern generation schemes is largely devoted to developing heuristic methods. Test pattern generation for a single fault has been shown [Iba75] [Fuj82] to belong to a class of problems known as "NP-Complete" [Gar78], which are believed to inherently require an exponential number of steps to solve in the worst case, written $O(2^n)$,* where n refers to the size of the program input, in this case the

* A function $f(n)$ is said to be $O(g(n))$ iff $f(n) \leq kg(n)$ for all $n \geq n_0$, where k and n_0 are constants

number of circuit inputs. Thus, adding a single input to a circuit can potentially double the amount of time required to generate test patterns for it. The source of the complexity is *reconvergent fanout*. Lines called fanout stems branch and then later reconverge, causing correlation between gate input values

2.3.2 Random Test Pattern Generation

Selecting test vectors at random has been proposed as a means of reducing the costs of deterministic test pattern generation. The original use of random test stimuli in automated test pattern generation was in *fault insertion*. Fault insertion [Sus73] involves physically inserting failures in a replica of the circuit to be tested, then running various inputs and noting when responses change. This of course requires physical access to the potential fault sites. There is typically no such access inside an IC * Fault insertion for ICs requires the circuit to be modelled by a collection of simpler packages, usually TTL chips, each containing only a few transistors and implementing a small part of the circuit. A circuit such as Intel's i860 RISC processor [Per89] contains over 1 million transistors. The logistics of producing a fault-free TTL version, let alone inserting faults into it, are difficult to comprehend. Furthermore, important factors such as circuit timing often cannot be duplicated in a large-scale TTL model For this reason, physical fault insertion is no longer a practical method of developing test sets. The concept remains, however. in random test pattern generation (RTPG) [Bre71] [Agr72] [Agr75]

RTPG mimics fault insertion by using *fault injection* coupled with *fault simulation*. A list of faults is generated using some fault model. These are injected into a circuit description whose behaviour is then simulated over a set of randomly selected input vectors This simulation is used to see which, if any, faults are detected by each successive pattern, and only those patterns which detect a previously undetected fault are retained The operation can terminate when a desired level of fault coverage has been achieved, or once a limit on the amount of effort to be expended has been exceeded Despite a recent claim that RTPG has no value [Abr89], there remains a class of faults for which deterministic tests are difficult to generate, yet can be detected easily

Thus $f(n) = 3n^2 + 2n + 4$ is $O(g(n))$ where $g(n) = n^2$ since $f(n) \leq 10g(n)$ for all $n \geq 1$

* Electron beam testers (see for example [Yan87] [Lee89]) offer physical observation of surface sites on a silicon chip, but do not offer the potential to temporarily insert failures into a working circuit One manufacturer now offers individual site access as part of its gate array design [Swa89], but such efforts are rare and probably excessive

by randomly-selected patterns. ATPG is often accomplished by using RTPG initially, followed by DTPG on the remaining undetected faults.

2.3.2.1 Fault Simulation in RTPG

In simulation, circuit behaviour is determined algorithmically from a circuit description. Simulation of a fault-free circuit is known as *circuit simulation*, while simulation where a fault has been injected into a circuit description is called *fault simulation*. Simulators have progressed from rudimentary parallel fault simulation [Ses65], where all faults are simulated in parallel for a given input pattern, through deductive simulators [Arm72] and the related concurrent simulators [Ulr74], where divergences between faulty and fault-free values are deduced logically, to parallel pattern single fault simulators [Wai85], where the word length of the simulating processor is used to obtain a reduction in the simulation time, to those employing stem regions [Maa88] [Maa90], which take advantage of circuit structure in order to speed execution.

2.3.2.2 Complexity of Fault Simulation

Simulating one fault for one input pattern is believed to require $O(G^2)$ steps in the worst case [Goe80] [Har87],† where G here refers to the number of gates within a circuit. As in section 2.3.1.2, the source of the complexity is reconvergent fanout while doubling the number of circuit gates can quadruple simulation time.

While this worst case complexity is retained by all simulators, actual simulation time can vary immensely between methods. The use of heuristics, parallelism, and circuit information all aid in reducing execution time. The fastest simulators are those which are designed exclusively for combinational circuits and stuck-at faults [Wai85] [Maa88]. An expanded fault model or simulation over sequential circuits requires other algorithms, such as those listed above, which tend to be slower in combinational simulation.

Fault simulation has become so expensive that special purpose computers, called *hardware accelerators*, have been developed for the express purpose of performing simulations. Overviews of these are given in [Bla84] and [MuE90].

2.3.3 Deterministic Fault Location Techniques

The ATPG techniques discussed so far have tended to emphasize fault detection as

† It is shown in [Har87] that if fault simulation could be performed in linear time ($O(G)$), then the product of two $m \times m$ matrices could be obtained in $O(m^2)$ time. All common algorithms for this operation [Aho74] are $O(m^3)$ complexity. No known algorithm is $O(m^2)$.

a primary goal over fault location. This shift in perspective results from manufacturing considerations. In terms of producing a fault-free product, manufacturers are typically interested in locating failures down only to the smallest replaceable component, usually an IC [Sus73] [Bat76], while remaining able to detect any failure which may occur. Individual failure location within an IC is less important for immediate production and shipping of circuits than it is for improving circuit design and the manufacturing process itself by providing some feedback about weak points. Once these have been identified, corrective action may be taken and process yields can improve. Such feedback between the design and production divisions of a circuit manufacturer characterizes a "living process" according to Parker [Par87], who states that such processes are more profitable than conventional static processes.

Although much research effort is devoted to fault detection, fault location nonetheless remains an important topic. An important example of the need for reliable fault location is the Thermal Conduction Packaging technique, where large numbers of die are mounted on a single wafer [Blo82] The entire Thermal Conduction Module (TCM) behaves as a large combinational circuit during test mode [Cur83]. Fault location is required for both module repair (by replacing faulty silicon die) and for tracking repetitive failure mechanisms. Both problems can be solved simultaneously by locating faults to individual circuit elements [Cur83]. While TCM circuits have the complexity of boards, conventional board diagnosis methods cannot be applied to them. Fault diagnosis in such systems is a major motivation of this dissertation

Fault dictionaries have been proposed by many authors as reasonable tools for fault location However, dictionary methods can fail when a multiple fault model is used. Consider, for example, a circuit with L lines Neglecting fault equivalence, there are $2L$ possible single stuck-at faults within the circuit. In a multiple stuck-at fault model, every line can be in one of three states: fault-free, stuck-at 1, or stuck-at 0. The total number of multiple stuck-at faults is thus $3^L - 1$ This number is completely beyond comprehension for circuits with thousands of lines. Hayes [Hay71] showed that fault equivalence reduces this number to less than 2^s, where s is the total number of inputs to all the fanout-free subsections of the circuit, but even this number is usually far beyond enumeration ability.

Thus, no fault dictionary can be constructed to include all multiple faults. Since multiple faults can mask the behaviour of single faults, a single fault dictionary, even if lists every failing pattern for every fault, may not always be useful. When multiple faults can occur in circuits, a fault location technique should provide some means of dealing with them. A variety of such methods have been proposed.

Some authors have attempted to determine multiple fault coverage through enumeration, for example by solving boolean equations resulting from test data [Bos71]. Eventually, however, the sheer number of faults will render such techniques unusable. It is possible to reduce the complexity by ignoring faults with greater than a certain multiplicity. This approach has intuitive appeal, since circuits containing more than a few faults have failed catastrophically — knowing which lines are responsible may not be overly beneficial. Of course, the definition of "few" will depend on the size of the circuit, its fault-tolerance, and the frequency with which such catastrophic failures occur. A problem with any such explicit enumeration is that the number of faults in the dictionary is likely to be far greater than the number actually observed in practice during the circuit's lifetime. In a sense, the effort expended to create the unused entries has been "wasted" [Abr80].

Another approach is that advocated by Abramovici and Breuer [Abr80] and more recently by Rajski and Cox [Raj87], of implicit multiple fault diagnosis by identifying lines which can be shown to be fault-free. The faults are then isolated to the remaining lines. The basic premise of the methods is that any line which can be shown to have had a transition is known to be neither stuck-at 0 nor stuck-at 1. The CERBERUS system of [Raj87] extends the idea to include multiple stuck-open and delay faults, by noting, for example, that if a $0 \rightarrow 1$ transition can be shown to have occurred, then the line cannot have a slow-to-rise fault. While the methods of both [Abr80] and [Raj87] are pessimistic, in that they may not locate all fault-free lines, they are able to give a set of lines which is known to contain any modelled faults which have occurred.

All the methods in this section are fault model dependent. Most employ only the multiple stuck-at fault model. CERBERUS [Raj87] adds stuck-open and delay faults to its basic fault set. None of the models includes bridging faults (since any line could be bridged to any other, the number of multiple bridging faults could be, depending on symmetry and transitivity constraints, as great as $O(2^{(L-1)^2})$) or intermittent faults, whose presence could invalidate conclusions as to presence or absence of faults on given lines.

The use of layout-dependent fault models may permit a single fault model to be retained. This is the approach of Inductive Fault Analysis [She85], proposed by Shen et al. as an alternative to the layout-independent fault models noted previously. In this technique, silicon point defects are simulated to determine their effect on the electrical characteristics of their surroundings. Those which cause faulty behaviour are retained in a fault list. This list is collapsed to eliminate as many duplicates as possible. The number of defects which collapse into a particular faulty behaviour is used as an indicator of the relative probability of that failure occurring. Since single point defects can manifest themselves as multiple stuck-at or stuck-open failures, Shen et al. claim that their single defect analysis is sufficient for most failures. A fault dictionary could be developed using this method, providing a circuit-based and thus potentially more accurate diagnosis tool.

Expanding the fault model permits diagnosis of more potential faults, but at an increased cost. Tradeoffs betw^en the costs of fault location and the benefits of increased potential resolution must be evaluated. In some cases, a simpler fault model with fast, inexpensive diagnosis may be preferable to a more costly, but complete, model, while in other cases the reverse may be true.

2.3.4 Testing Equipment

As circuit complexity has increased, so has the complexity of circuit testing equipment. Testers have become more and more expensive as they have progressed from *ad hoc* devices assembled for a given circuit design, to internally developed general testing devices, to a wide variety of commercially available models which now form their own industry. A good history and general description of these is available in [Par87]. Circuit testers suffer from two major problems: expense and performance. Testers can cost millions of dollars, yet can never match the state of the art in component speed, since the tester is an "old" design by the time it reaches the market. The performance factor is usually offset by differences in device technologies. For example, CMOS, a common device technology, tends to be much slower than ECL, so CMOS testers are often implemented in ECL.

2.4 Use of Exhaustive Testing in Fault Diagnosis

An exhaustive test is guaranteed to detect all combinational faults in a circuit (not necessarily sequential faults, such as stuck-open or delay faults) For this reason, some authors, e g [Sed79] [McC81], have suggested the use of these test sets in fault diagnosis.

Since exhaustive tests are not feasible for circuits with more than about 20 inputs, partitioning techniques to allow exhaustive excitation of subcircuits have been developed. Some examples of these pseudo-exhaustive techniques may be found in [Boz80] [McC81] [Ude88]. The bibliography in [Ude88] provides a good overview of research efforts in exhaustive and pseudo-exhaustive testing.

An exhaustive or pseudo-exhaustive (also known as a *verification test* [McC87]) can require extremely long test lengths, on the order of millions of test vectors. As a result, some form of output compaction is required. Methods of performing this compaction are investigated in section 3.5. Exhaustive testing avoids the costs involved in determining the quality of test sets and allows for simpler testing equipment than ATPG, but at the expense of additional test time.

2.5 Use of Random Vectors in Fault Diagnosis

Selecting patterns at random has been proposed as a way to avoid both the expense of generating deterministic test patterns and the exponential test lengths of exhaustive testing [Bre71] [Agr72] [Bas73] [Sus73] [Agr75] [Dav76]. In most cases patterns are selected in some pseudo-random fashion,* both for ease of generation and in order to avoid repeats. Two common pseudo-random generating techniques are linear feedback shift register (LFSR) sequences [Pet72] [Gol67] and cellular automaton (CA) sequences [Wol83] [Hor89], each of which can generate, in a deterministic but apparently random order, a maximal sequence of input patterns [Pet72] [Ser88].† Two options are available when random or pseudo-random input patterns are employed: One, RTPG, retains only those patterns which detect a previously undetected fault and has already been examined (see section 2.3.2); while the other applies all patterns and provides the basis for *random pattern test.*

2.5.1 Random Pattern Test

In random pattern test (RPT), the test is made sufficiently long to detect a high percentage of faults with high probability. (Savir et al [Sav84] suggest detection of 98% of the faults with 99.9% probability. The confidence level comes from the use of testability measures, which are discussed in section 2.5.3.) Fault coverage of individual patterns is not determined, resulting in a reduction in fault simulation cost at the expense of added test application time. When an LFSR or CA is used to generate the input patterns, some of the complexity of standard testers may be avoided. Without simulation, however, the fault coverage of the test set can only be estimated, no fault

* Pseudo-random vectors are able to pass some tests for randomness, but are characterized by the repeatability of the sequence, and often by a lack of repetition in the patterns themselves.

† A maximal sequence of n-bit vectors is of length $2^n - 1$, since the all 0 pattern forms its own sequence. Adding the all 0 pattern to a maximal sequence produces an exhaustive sequence. Some methods of adapting LFSRs to produce exhaustive sequences are given in [McC86] and [Wan86].

dictionary can be constructed, and fault location cannot be performed. (Simulation over a fraction of the fault set has been proposed as a means of estimating coverage with reduced effort [Jai84]). These random pattern test sets tend to be longer by several orders of magnitude than their deterministic equivalents [Sus73] [Chi87], although this additional length may allow them to detect some unmodelled faults that would be missed by a shorter deterministic test set. Recent experiments by Maxwell [Max89] have documented the existence of such faults in actual manufactured circuits.

When RPT is used, several choices are available for analysis of the circuit output. The results may simply be compared against the fault-free response (from simulation), a reference or *gold unit* may be used [Dav76] [She77],* or finally the output may be compacted into a more manageable size [Los78]. This last alternative is known as *random compact testing.*

2.5.2 Random Compact Testing

Random compact testing (RCT) employs data compaction, a process which records some attribute (or set of attributes) of an output sequence. This attribute is known as the sequence's *signature*, and is typically orders of magnitude smaller than the original sequence, e.g., a 32 bit signature for a sequence several million bits in size.

The RCT environment consists of a psuedo-random input generator, the circuit under test, a response compactor, and a comparator to match the observed signature with its expected (fault-free) value. This set-up is shown in figure 2.2.

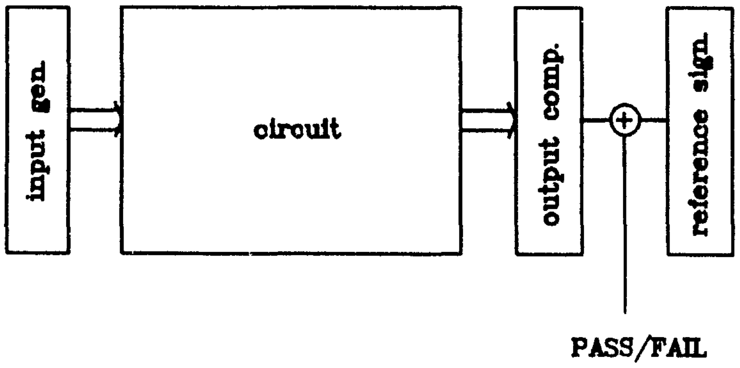


Figure 2.2 The RCT Environment

The input generator is typically an LFSR, but potentially a cellular automaton (CA) or similar structure, or may be a pseudo-exhaustive structure, such as a counter or

* Of course, the use of a reference unit begs the question of how such a device is known to be fault-free

maximal length LFSR or CA. The RCT environment is also used in another technique, referred to as built-in self-test. More information on data compaction techniques is given in section 3.5.

2.5.3 Fault Detection Probability

When testing is performed with randomly or pseudo-randomly selected test patterns, test length is determined by the *detection probability* of the faults in question. This probability can be defined for a given fault f as:

$$p_f = P(\text{Randomly selected pattern detects } f)$$

If each pattern has equal probability of being selected, the fault's detection probability is thus:

$$p_f = \frac{N_f}{2^l}$$

where N_f patterns detect the fault and l is the number of circuit inputs.

The test length required for a random test can be determined from a fixed non-detection probability threshold P_{nd} for the "hardest" fault — the one with the smallest detection probability. David [Dav76] suggests choosing test length, n, so that the "detection uncertainty", P_{nd} or the probability that the hardest fault within the circuit will escape detection, of the circuit is reduced to some desired value. Thus, given

$$p_{min} = \min_{\text{all } f}(p_f)$$

and a fixed threshold P_{nd}, test length can be determined from

$$(1 - p_{min})^n \leq P_{nd}$$

so

$$n \geq \frac{\log(P_{nd})}{\log(1 - p_f)}$$

The use of pseudo-random, rather than random, vectors complicates the problem somewhat [Chi87] [Wag87], although the overall test length is often reduced

Detection probability is important in RTPG as well, since the test length in RPT is roughly equivalent to the number of patterns likely to be simulated before a complete test set is found. Detection probability is one of numerous *testability measures* which have been proposed, e.g. [Gol79] [Brg84] [Set85] [Kri86] [Set86]. Testability measures are intended to give some idea of the amount of effort which will be required for test generation, without actually generating any test. Circuits which are labelled difficult to test may then be redesigned in order to make them more easily testable. As has been pointed out in [Agr82] and [Sav83], testability measures may be of limited use in deterministic test pattern generation, but they remain important in all facets of random testing (one testability measure, *test counting* [Ake89], is claimed by its authors to be more useful in deterministic test generation than in random testing). Most of these measures can be converted to detection probability in a straightforward manner, so no further distinction will be made between them.

Calculation of exact detection probabilities for each fault is a time consuming operation. It has in fact been shown to be a #P-complete problem [Kri86].* The method described in [Par75a] [Par75b] gives exact results, but a fault may require exponential time in the worst case.

The difficulties involved in exact calculation of these probabilities have led to a variety of approximate methods. Reconvergent fanout again is the source of the complexity, (without it, computing detection probabilities is a linear algorithm in the number of circuit lines, L), and is handled in different ways by different algorithms. The simplest, COP [Brg84], ignores it entirely, and has $O(L)$ complexity as a result. The algorithm in [Kri86] uses repeated applications of a slightly modified COP strategy to eliminate the first-order effects of input reconvergence. Other algorithms attempt to eliminate the reconvergence through the use of "supergates" [Set85] [Set86], which treat fanout regions as individual gates, perform exhaustive analysis on them, and use the linear algorithm everywhere else. Such algorithms give exact results but retain the worst-case exponential time complexity. The "cutting algorithm" in [Sav84] provides bounds on detection probability, although these can be very weak [Gae86]. Statistical methods have also been developed, whether with fault simulation [Wai85] [Bri86], or without [Jai84]. More comprehensive reviews of these methods may be found in [Hui88] and [Pat90].

Throughout this dissertation it is assumed that random-pattern detection probability, whether exact or estimated, is available for each fault. No attempt is made to add to the wealth of information already available on its calculation

* Similar to NP-complete problems, only the number of solutions, rather than the existence of a solution, is sought [Gar78] Since the number of solutions may be exponential, guessing these in one step would still not help, as the time required to verify them would remain at least $O(2^n)$ using known algorithms

2.5.4 Random Pattern Resistant Faults

There are certain types of faults which have very small detection probabilities, and hence require long random test lengths. These are known as *random pattern resistant* faults. Totton and Shaw [Tot88] give four main causes of random pattern resistance

- Redundancy: These faults cannot be detected by any test pattern, and hence have detection probability 0.
- Reconvergent fanout: Fault effects on reconvergent paths can cancel one another out.
- High fan-in: It may be difficult to set values which will permit propagation
- Test vector quality. Correlation between successive patterns, especially from an LFSR, may make detection more difficult.

Since random pattern test length is usually determined by the lowest detection probability in the circuit, random pattern resistant faults are a serious problem One potential solution is biasing the random selection algorithm.

2.5.5 Weighted Random Patterns

Detection probability of random pattern resistant faults, and consequently the length of a pseudo-random test sequence, can be altered through the use of biased or weighted random patterns [Sch75] [Sav84] [Wun87] [Wai88] [MuF90]. These change the selection probability of each input vector in an effort to make the hardest faults more easily testable. The weighting algorithms typically focus on individual inputs, giving a fraction of the time for which each input will be 1 (this fraction will be $\frac{1}{2}$ for an unbiased input). These biases can be used by the methods of section 2.5.3 to obtain estimates of detection probability.

Weighted random patterns offer other advantages in addition to shorter test lengths, notably the potential to detect faults more frequently than they would be detected by an unweighted set. Varying degrees of success have been reported recently [Wun87] [Wai88], and although to date no provably optimal strategy for selecting the biasing method has been reported, the use of biased vectors can reduce random test lengths by several orders of magnitude [MuF90]

2.5.6 Fault Location with Random Vectors

Fault location when pseudo-random test vectors are employed has long been considered a difficult problem [McA87]. This dissertation proposes a solution, which is discussed in later chapters. A brief overview of the problem is given here — a detailed review of previously proposed techniques for fault location in the random compact testing environment is given in chapter 4.

When simulation is not performed to validate the performance of a test set, no dictionary is generated. Creating one is an expensive process, again with effectiveness and cost factors in the choice of fault models. Faults can be diagnosed by performing circuit analysis or simulation after the fact in an effort to construct the dictionary entries for the fault(s) observed [Arz81]. Without excellent search techniques, there will be little saving in simulation time over the initial generation of a fault dictionary for the entire fault set and all input patterns, especially if data compaction is used. Whether the fault dictionary is constructed in advance, or in a demand-driven fashion, fault simulation is the largest cost component in fault location in the RCT environment.

2.5.6.1 Complexity of Fault Simulation

The major expense in generating complete fault dictionaries for use with random testing is fault simulation These dictionaries, especially in RCT, require the generation of every output bit for every fault. As mentioned in section 2.3.2.2, fault simulation is believed to inherently require $O(G^2)$ steps, where G is the number of gates in the circuit, for one input vector and for one fault. For an n vector test on an m output circuit, the complexity for h distinct faults and n input vectors is certainly not less than $O(hmn)$. This requirement of full fault simulation to produce complete output functions eliminates many of the "tricks" used by fault simulators in an effort to reduce simulation time.

For example, one of the fastest combinational circuit simulation methods available is that used by Tulip [Maa88] [Maa90]. When the simulator is used for RTPG, time is saved by fault-dropping, or eliminating faults from consideration after they have been detected once. Additional savings are obtained by dropping fault-free simulation of entire regions of the circuit, once all the faults within them are detected None of the gains from either of these is available when the faulty output function must be produced, since each fault must be simulated to each output for each pattern

In combinational fault simulation, a fixed speedup related to the machine word length may be obtained by simulating several patterns simultaneously for a given fault [Wai85]. Because this speedup is by a constant factor, however, the complexity of the operation is not reduced Further, the bits must be "unpacked" from the word in order to produce a usable output function for signature calculation.

In order to make the results in this dissertation relatively independent of the fault simulation tools used in actual diagnosis, the following assumption is made

Assumption 2.1 If the effort required to simulate a given circuit for one input vector in order to produce one output vector for one fault is x, then the simulation effort required to produce k output vectors for one fault is kx and the effort required to produce one output vector for each of h faults is hx.

This assumption is an approximation of reality for some simulators, but it provides a good basis for comparison of effort. Further, its accuracy is likely to improve when sequential machines such as signature analyzers are included in the simulation, precluding parallel-pattern methods for at least part of the circuit. The assumption ignores the start-up and overhead costs of simulators, but if the simulation tools were being used frequently enough, they could be adapted so as to reduce these costs (by remaining in memory, for instance).

2.6 Design for Testability

The expense of test pattern generation and fault simulation, coupled with the increasing cost of circuit testers, has left manufacturers with two choices: risk shipping more defective products, or attempt to lower the costs through some form of design for testability (DFT) [Wil79] [Wil83].* Whether these techniques are *ad hoc* suggestions such as reducing or eliminating wired-OR configurations, or a rigid set of design rules, such as those imposed by IBM for level-sensitive scan design (LSSD) [Eic78], their goal is to reduce the costs of fault diagnosis. A recent definition of design for testability given by Agrawal [MuE90] is much more restrictive, stating that anything added after the core of a design is complete, including built-in self-test and scan methodologies, cannot be considered part of design for testability. This latest definition might perhaps be better termed synthesis for testability, and this dissertation will use the original, broader definition.

Of the structured DFT techniques, scan design is probably the best-known. Most fault diagnosis methods apply to combinational circuits, while the majority of actual designs are sequential. Scan design converts the sequential elements of a circuit (latches or flip-flops) into primary inputs and outputs for the purpose of testing. This has the effect of changing a sequential circuit into a purely combinational one for test purposes, and allows direct observation and control of the internal state of the circuit.

* Williams' and Parker's survey provides an excellent starting point for any study of DFT

A variety of scan techniques have been proposed, e.g. [Wil73] [Fun75] [Ste77] [Eic78] [And80], (overviews in [Fuj85] [Lal85] [McC85]), each with its own design rules and latch designs, but the overall principle remains roughly constant. A good history of scan design techniques is available in [Fun89] Some examples of scan methods are LSSD [Eic78], scan/set design [Ste77] and random access scan [And80]. Of these, some form of LSSD is perhaps the most common.

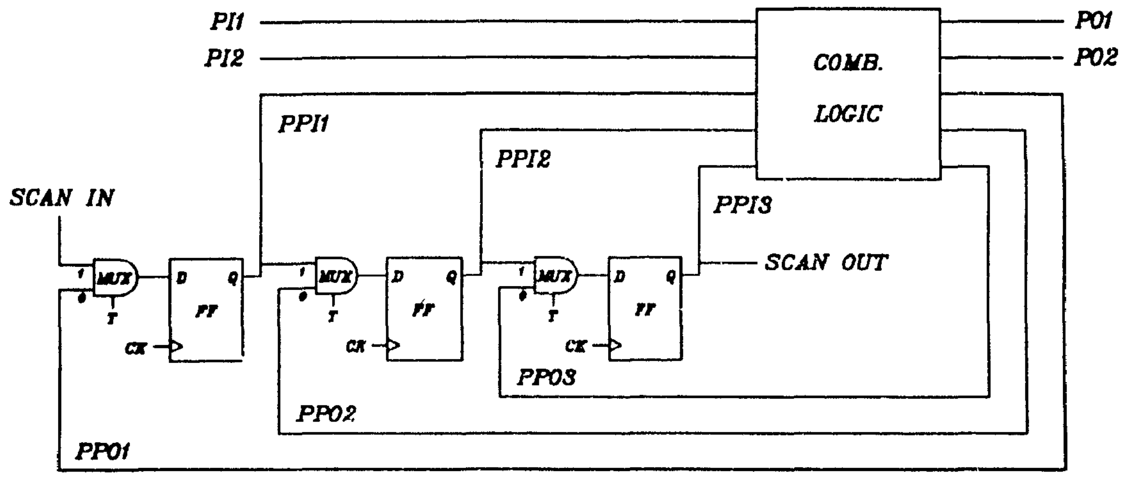


Figure 2.3 Typical Scan Design

A typical scan implementation is shown in figure 2.3.[†] There are two modes of circuit operation in a scan design, normal and test. In normal mode, the additional scan hardware is transparent, and the circuit behaves normally. During test mode, input vectors are serially loaded into the latches via the scan chain. The circuit is then permitted one cycle of operation in normal mode, and then the output vectors are shifted serially out through the scan chain and observed

The use of scan design requires that the sequential elements be redesigned to permit the two modes. This results in additional silicon area overhead for the test mode interconnect (including scan chain and test clock(s)), the more complex latches, plus an additional multiplexer at the input to each latch so that input values may be selected from either the combinational logic or the scan chain. This multiplexer also adds to power consumption and introduces a time delay into the circuit's normal mode which, while not severe, can be significant enough to slow down a high-performance design. Finally, some functional testing is necessary, since timing faults cannot usually be detected by a scan test These overheads continue to deter some companies from using

[†] Figure courtesy of Fadi Maamari.

structured DFT over *ad hoc* approaches [Fel89].

One form of DFT which is gaining in popularity is built-in self-test (BIST) where a circuit is responsible for testing itself. These self tests are described in chapter 3

2.6.1 Board-Level Approaches

Scan techniques such as LSSD operate at the chip level In order to take advantage of some of the benefits of scan design at the board level, *boundary scan* has been proposed (see, for example, [Mau87] [Glo88] [Has88]). In a boundary scan, the inputs and outputs of every chip on a board can be configured into a scan chain for test purposes The boundary scan chain can bypass a given chip for all or part of the test The practical implementation of such a technique requires a certain amount of standardization in chip architecture among manufacturers This consensus has been sought by a committee known as the Joint Test Advisory Group (JTAG), and is likely to result in IEEE standard P1149.1 [Fit90].

A boundary scan provides for a hierarchical design-for-testability approach As board complexities increase, the controllability and observability of chip inputs decreases. Conventional approaches (see [Par87] [Gra89]), such as bed-of-nails test fixtures together with overdriving of components, are no longer applicable Surface-mount technology blocks access to a bed-of-nails tester, and overdriving can damage devices when test times are long. Boundary scan eliminates these problems and helps to provide an integrated test scheme for testing boards whose component parts are supplied by different vendors.

Not all research is directed toward such structured techniques. *Ad hoc* approaches for board-level tests continue to be suggested [Tur89] [VaN89] as alternatives to the potentially more costly and still unfinalized boundary scan methods These are typically suggested as “stop–gap” measures, to be used until boundary scan is universally available.

Chapter 3 Built-In Self-Test and Data Compaction

An important class of design for testability methods involves incorporating test circuitry on the chip itself. In such Built-In Self-Test (BIST) techniques, a chip tests itself and reports the results of the test to the outside world. There are two major types of BIST: programmed and structured In the first, a ROM stores a program for generating tests and analyzing responses, while in the second, standardized circuit structures for test pattern generation and output response analysis are included on-chip. A third possible method, where input and expected output vectors are stored on-chip in ROM has never been investigated in depth because of the large hardware overhead anticipated in most cases.

Self-testing offers numerous advantages over conventional approaches, including:

- The complexity of external test hardware can be dramatically reduced over conventional methods, allowing a corresponding cost decrease.
- Longer test lengths can be run efficiently.
- The random vectors often used in a self-test have the potential to detect many unmodelled faults
- Tests can frequently be run at closer to circuit speeds (although not in scan designs).

Such benefits are not achieved without cost. The additional area overhead of the BIST circuitry can lower yield. In addition, the test circuitry can reduce circuit performance if it is present in critical paths, and the fault coverage obtained with BIST is not always known None of these problems is insurmountable, however, and BIST is emerging as a useful method in fault diagnosis [Bar87].

The hierarchical method of fault location proposed in this dissertation is able to alleviate a final problem with the data compaction techniques used by both BIST and random compact testing (see section 2.5.2): Fault location has been difficult, if not impossible, in signature-tested circuits [McA87]. The proposed hierarchical method is described in later chapters. First, however, a review of BIST practices and proposals is given.

3.1 Programmed BIST

Programmed BIST is usually applied to microprocessors, although store and generate BIST [Aga81] [Bar87] may be thought of as a programmed method. A small test program can be stored in ROM This program, when run, will generate test vectors and analyze responses, comparing them with expected values. Such self tests are becoming common [Kub83] [Gel86] [Fel89] [Har89] as part of the manufacturing process for microprocessors, and have been made accessible to the user to allow additional testing during the microprocessor's lifetime [Kub83]. Examples of functional test programs for microprocessors can be found in [Bra84]. The fault coverage and diagnostic resolution of programmed BIST depends on the quality of the test program. Programmed BIST will not be considered further in this dissertation.

3.2 Structured BIST

Structured BIST uses the random compact testing environment described in section 2.5.2 and shown in figure 2.2, with all hardware (input generator, output compactor and comparator) located on the chip or board with the circuit under test (CUT) itself Because of their similar environments, many of the results of the following sections also apply to random compac testing. BIST has been proposed for both regular structures, such as memories [Sun84] and PLAs [Tre85] [Ser87], and so-called random logic. This dissertation investigates only the latter. The methods given may be applied to structured circuits, but only the more general circuits are considered explicitly.

3.3 Performance Measures of BIST

Because of the large number of test vectors required by BIST techniques, full fault simulation may not be performed. As a result, actual fault coverage of the methods frequently remains unknown Instead, the techniques are often compared in terms of their *aliasing* or *masking* probability — the probability that the signature of a faulty circuit will be identical to that of the fault-free circuit While fault coverage in most testing schemes is a monotonically increasing function, the information loss possible in

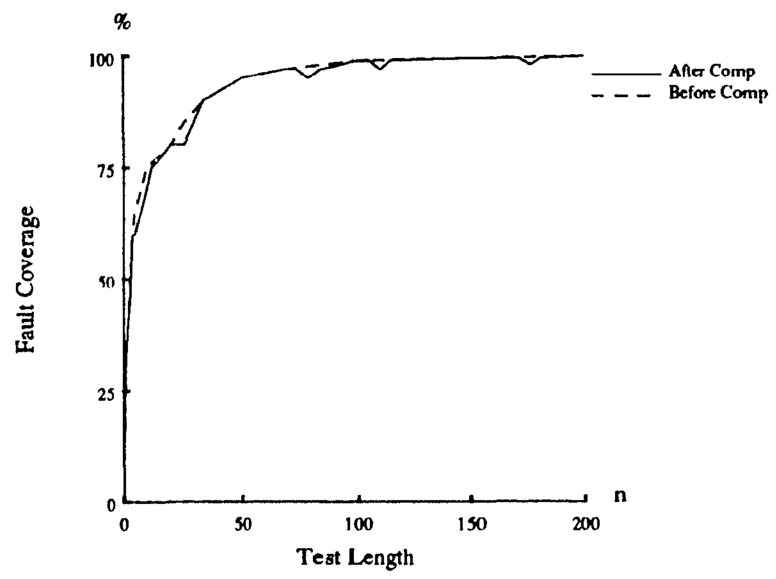


Figure 3.1 Fault Coverage Before and After Data Compaction

data compaction means that adding another test vector can potentially reduce fault coverage, as shown in figure 3.1.

Performance measures for BIST have been divided into two classes [Iva88b]: Those based on fault coverage and those based on error coverage In the former, the actual behaviour of faults within some fault model are considered, while in the latter faults are ignored in favour of some means of modelling the behaviour of fault effects or errors in the output vectors. These models are known as *error models* and are the basis of results in aliasing probability. Examples of performance measures based on fault coverage can be found in [Sav80a] [Mar81] [Mil83] [Mil84], while some based on error coverage may be found in [Fro77] [Smi80] [CaJ82] [Aga83] [Has84] [Sav85] [Wil86] [Muz87] [Dam88] [Gup88] [Iva88a] [Wil88] and [Ait88a].

In addition to mathematical predictions of aliasing, fault coverage measures may also be obtained by simulation. Examples of simulation experiments to determine fault coverage may be found in [Ait86] [McC87] and [Xav89]. Experimental analysis is also helpful for determining the validity of error models. Without such information, confidence with regard to aliasing behaviour is limited

3.4 Error Models

A variety of error models have been proposed for modelling faults and their aliasing affects with various BIST techniques. Most of the proposed models have their origins in coding theory (see for example [Pet72] [Sha48]). Some authors have gone so far as to eschew error models entirely [CaJ82], but most have used one of those given below·

3.4.1 Uniform Error Model

The original and simplest of the error models, the uniform model assumes that in the presence of a fault, every output pattern has precisely the same chance of occurring. While this model makes for straightforward calculations [Fro77] [Smi80] [Sav85], its connection with reality is limited at best The use of this model implies the denial of any circuit information whatsoever, which seems unlikely to increase the accuracy of performance projections.

While some authors have supported the uniform model [Los78] [Hsi84], others have argued against it since its inception [Smi80] [CaJ82] [CaW82]. Experiments questioning its validity have been performed [Ait86], but the model's use appears to have been perpetuated by the lack of a better alternative.

3.4.2 Burst Error Model

Derived from coding theory, where it was meant to model bursts of noise on communication channels, the burst error model has received some attention in the BIST literature [Smi80] [Sax87] [Iva88b]. In the context of BIST, a burst error is defined as an error sequence where some dependencies exist in the positioning of erroneous bits, which are usually restricted to positions spaced by some power of 2. For such dependencies to exist on the outputs of a circuit, they must clearly also exist on its input stimuli, so the burst error model is probably only applicable to cases where circuit inputs are generated by a counter, rather than pseudo-randomly by an LFSR or a CA. Since the fault location method described in this dissertation requires some randomness in its input generation, a counter is not an appropriate input device, hence burst errors will not be considered further.

3.4.3 Independent Error Model

The independent error model, first applied to BIST in a general way by Williams et al in [Wil86], although David used it in a specific case in [Dav80], is based on the binary symmetric channel in coding theory. Errors in an output stream (responses at a given output of a circuit) are assumed to occur randomly and independently with some fixed probability p. This p is equivalent to the random pattern detectability of the fault (section 2.5.3), so the independent error model presupposes random, or at least pseudo-random, input patterns.* Note that the uniform model may be considered a special case of the independent model where $p = \frac{1}{2}$ for all faults.

The independent error model takes some circuit information into consideration, and hence intuitively looks more promising than the uniform model in terms of its performance. Experiments reported in this dissertation and others in [Xav89] tend to confirm this improvement, although the model is still not without its performance flaws [Ait88b] [Ait89b].

3.4.4 Asymmetric Error Model

The asymmetric error model is an extension of the independent error model in the same way as the binary asymmetric channel is a generalization of the binary symmetric channel. While communication channels may be physically altered to eliminate asymmetric behaviour [Pet72], such freedom is not readily available when developing random tests for a given circuit. The asymmetric error model for BIST was first described in [Ait88b] and may be summarized as follows:

As noted previously, the independent error model employs the probability, p, of detecting an error on a given circuit output. In the asymmetric error model, this probability varies, depending on whether an error is a $1 \rightarrow 0$ change (fault-free to faulty) or a $0 \rightarrow 1$ change. These two changes were characterized as D and $\overline{D}$ respectively in [Rot66], the original reference on the D-algorithm.

Let event e denote the occurrence of an error on output line x which has fault-free value X. The random pattern detectability of a fault, p, may then be written as:

$$p = P(e) = P(e|X=1) \cdot P(X=1) + P(e|X=0) \cdot P(X=0)$$

The asymmetric probabilities are defined as:

$$p_D = P(e|X=1)$$
$$p_{\overline{D}} = P(e|X=0)$$

Hence

$$p = p_D s_X + p_{\overline{D}}(1 - s_X)$$

* Some comments on the differences between the use of random patterns (sampling with replacement) and pseudo-random patterns (sampling without replacement) may be found in [San60] and [Chi87]. The effects of these differences on BIST are discussed in [Ait88c]

where s_X is the fault-free signal probability of line x. The independent error model assumes that $p_D = p_{\overline{D}}$ and thus may be viewed as a special case of the asymmetric error model.

3.4.5 Generalized Error Model

The *generalized error model* allows each bit of an output sequence to have a unique error probability, say p_i for the ith such bit ($1 \leq i \leq n$). The asymmetric error model and hence also the independent error model are special cases of this model. The model was used in [Iva88b] in the development of aliasing probability for signature analysis. Clearly, the assignment of individual error probabilities is the most daunting task when using this model.

3.5 Data Compaction Techniques

Data compaction refers to the process of obtaining a signature from an output data sequence. The signature is typically smaller by orders of magnitude than the output sequence: e.g. a 32 bit signature for several million output bits. The process has also been referred to as *data compression*, but since the signature must contain less information than the original output sequence, data compaction will be used in this dissertation. A data compaction technique must satisfy several conditions to be useful in BIST or random compact testing (RCT): It must be implementable with a low hardware overhead, it must be compatible with DFT rules, and it must have low aliasing probability. A brief overview of some of the best-known data compaction techniques follows (additional information is available in [McC85] and [Bar87]):

3.5.1 Signature Analysis

The most common of all data compaction schemes, *signature analysis* is often considered to be virtually synonymous with BIST [Gra89] A signature analyzer, as defined by Frohwerk in [Fro77], is a polynomial divider implemented by a *linear feedback shift register* (LFSR) (see [Pet72] [Gol67]). The characteristic polynomial of the LFSR is called the *feedback polynomial* For a single output circuit, signature analysis treats the output stream as a polynomial, performs polynomial division over the Galois field GF(2) on this polynomial, and leaves the remainder in the LFSR as the signature. For multiple output circuits, a similar structure, known as a *multi-input signature register*

(MISR, pronounced "miser"), is used. Figure 3.2 shows an LFSR and a MISR.* The first proposal for a signature analyzer is given in [Ben75] Since then, numerous others have appeared, e.g. [Fro77] [Koe79] [Dav80] [Dav86] [Kra87]. A good review of the technique is available in [Bar87].

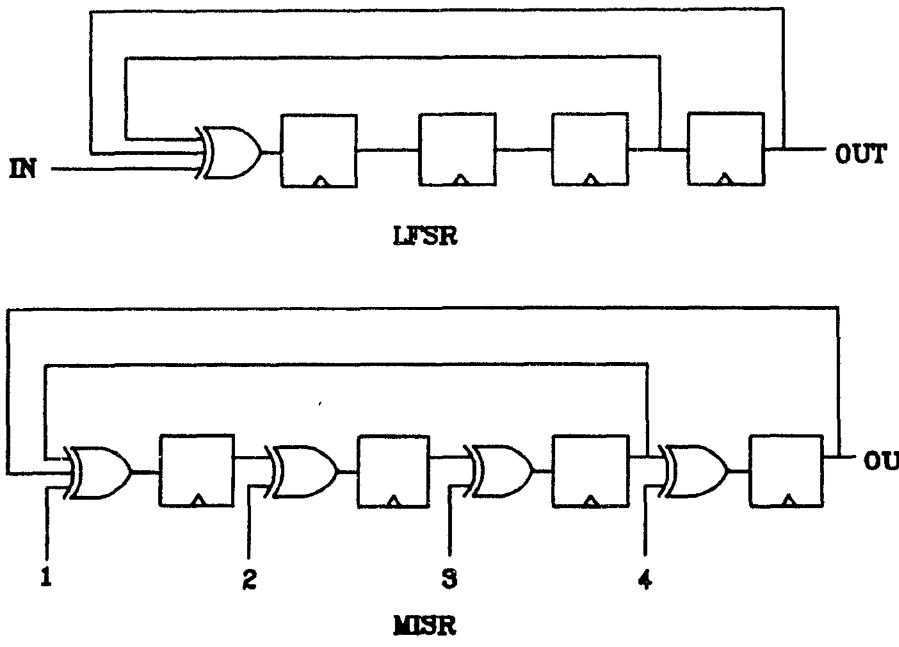


Figure 3.2 Signature Analyzers, Feedback polynomial: $x^4 + x + 1$

Signature analysis has been combined with scan design techniques (see section 2.6) to produce a structure known as a Built-In Logic Block Observer (BILBO) [Koe79]. A BILBO may be used as a test pattern generator, signature analyzer or scan register.

Another proposed BIST structure based on signature analysis is circular self-test [Kra87] [Kim88] [Pra88], where the same LFSR is used as both a pattern generator and signature analyzer simultaneously.

Polynomial dividers are not the only structures proposed for signature analysis. Another linear finite state machine, the *cellular automaton* [Wol83], has also been suggested [Hor88]. This would use a *Cellular Automaton Logic Block Observer* (CALBO) as a test structure. Linear cellular automata have been shown to have better randomness properties than LFSRs [Hor89] and are potentially better test pattern generators. Their behaviour as single-input signature analyzers is identical [Ser88], but recent results im-

* Figure 3 2 shows shift registers known as "type II" This type does not contain the precise remainder, but is isomorphic [Smi80] to "type I", whose exclusive-or gates are between the stages While type I LFSRs leave the exact remainder after division, type II are simpler to implement in hardware See [Smi80] or [Bar87] for more details

ply that multiple-input cellular automata (MICA) may have better aliasing properties than MISRs [Mil89]. Overhead differences between BILBOs and CALBOs are discussed in [Hur88].

3.5.2 Transition Counting

Transition counting, suggested by Hayes in [Hay76a], takes as a signature the number of 0-1 and 1-0 transitions on an output stream. Thus, the stream

0101011010111

contains 10 transitions, while

0111000111110

contains 4. Unlike the other methods described in this section, transition counting will not always detect a single-bit error. Very few results on transition counting are available; some of these are [Hay76a] [Red77] [Hay78] and [Sav85]. All of these works apply to transition counting on single-output circuits.

A method of fault location using transition counting is discussed in [Hay78], but the exponential test lengths required make it essentially impractical.

3.5.3 Weight Counting

Weight counting, also known as *one's counting* and *syndrome testing*, consists of summing the weight (number of 1s) in the output sequence. The term *weight counting* is introduced here to avoid the exhaustive test connotations associated with syndrome testing [Sav80a] [Bar81] [Mar81] [Sav81] as well as confusion with the coding theory definition of *syndrome* [Pet72]. An additional benefit is that test hardware for a weight count may be referred to as a "weight watcher". Most references to weight counting discuss only single-output circuits [Hay76b] [Los78] [Sav80a], but a generalization of syndrome testing to multi-output circuits has been proposed [Bar81] [Ser87]. This is weighted syndrome sums, where the signature is a weighted summation of individual output syndromes. The hardware overhead associated with this technique tends to be large, especially when compared with a MISR

3.5.4 Parity Testing

A variety of signatures based on the concept of *parity* have been proposed [CaW82] [Ake88] [Muz88] [Dam89]. Parity coefficients are based on the Reed-Muller canonical form of a function [Kum81] [Hur87] [Muz88] [Dam89], and it is possible to select a set of them which detects all faults within a given circuit, without resorting to an exhaustive test [Muz88] [Dam89]. Nonetheless, extensive computational effort is required to compute these sets [Muz88], although the method in [CaW82] suggests circuit modifications and exhaustive testing as an alternative.

3.5.5 Spectral Testing

Spectral testing encompasses both syndrome testing and parity testing. Excellent reviews of the subject are available in [Kar85] and [Hur85]. Spectral techniques map a boolean function into its equivalent in another coefficient system. The most frequently investigated has been the Rademacher-Walsh spectrum [Sus81] [Mil83] [Mil84] [Hsi84], although the Reed-Muller expansion (see the preceding section) and the Arithmetic expansion [Hei87] have also been employed. The interplay between the various spectra is described in [Hur87]. Because of the nature of the spectral coefficients, many signature techniques employing them require exhaustive tests. In addition, precise calculation of Walsh coefficients requires exhaustive simulation. Pseudo-random methods to calculate approximate coefficients have not been investigated in detail, with the exception of weight counting versus syndrome testing and the approximate technique of [Hsi84]

3.5.6 Output Data Modification

Output data modification (ODM), initially proposed by Agarwal and subsequently improved by Zorian, has been suggested as a method of reducing the aliasing probability of built-in self-test by orders of magnitude [Aga83] [Zor84] [Zor86] [Aga87] [Zor87]. The technique, a diagram of which is given in figure 3.3, combines signature analysis and weight counting in an interesting fashion. The multiple outputs of a circuit are compacted into a single-bit quotient stream by the MISR. This sequence is then exclusive-or'ed with a similar sequence, known as a modifier sequence and generated within the modifier block, which matches some fraction of the quotient stream bits. Various tradeoffs are possible between the degree of matching and the resulting aliasing probability, with better matchings yielding lower aliasing, but at higher overhead. The goal is to reduce the weight of the modified sequence as much as possible, since low weight sequences have smaller aliasing probability.

3.5.7 Other Techniques

Another BIST method which has been proposed is Accumulator Compression Test-

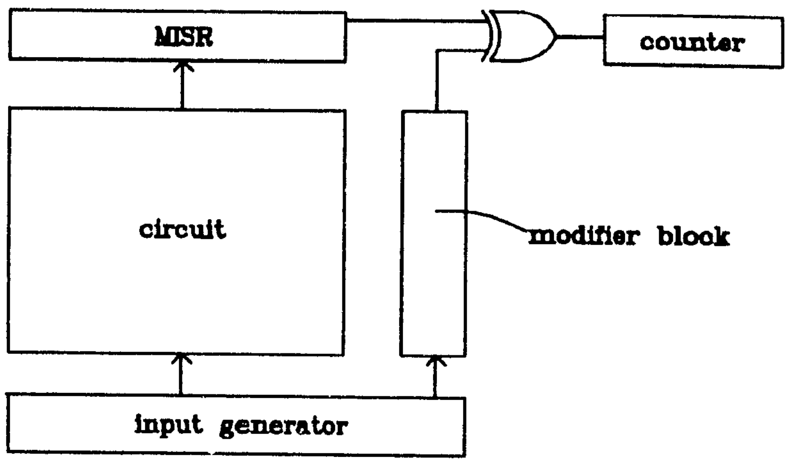


Figure 3.3 Output Data Modification Setup

ing [Sax86], where the weight count after each test vector is added in an accumulator to give a final signature. A performance analysis of the method is given in [Muz87] Other suggested methods include linear combinations of spectral coefficients [Muz87] and hybrid methods combining the signatures of one or more other BIST methods, e.g [Has84] [Rob87] [Rob88].

Chapter 4 Fault Location in Random Compact Testing and Built-In Self-Test

The most immediate problem in production testing is to determine which chips are faulty and which are not. This, together with the fact that random compact testing (RCT) and built-in self-test (BIST) techniques which employ data compaction are applied in most cases to a single chip, have resulted in a scarcity of research and results in the area of fault location properties of data compaction schemes. Nonetheless, there are good reasons to perform individual fault location when data compaction is used, for both RCT and BIST.

In any repairable system, fault location is clearly necessary. For example, a faulty module or board can be repaired by replacing faulty chips. Similarly, many fault-tolerant systems can be reconfigured to bypass faulty components, but reconfiguration cannot occur without fault location. Although these examples do not directly require resolution to an individual fault, when a fault has been isolated to a particular gate, all the hierarchical groupings of which the gate is a part are also known to be faulty. Finally, when a particular chip or subcircuit is found to fail frequently, the cause of this failure must be found before the problem can be solved. As was noted in section 2.3.3, Thermal Conduction packaging technology [Blo82] is a prime example of a system requiring good fault location.

The same reasoning applies to faulty circuits found during manufacturing test. Circuits with unacceptably low or even zero yields require design or process changes, and these in turn require fault location information. Such a failure analysis is important both during the start-up phase of production, and later in the life cycle, when analysis of failed chips from the field can provide information about weak points

This chapter reviews the current state of the art in fault location in the random compact testing environment, outlines the nature of the problem itself, and shows the limitations of previously proposed solutions. As with most of the data compaction literature, published results in fault location tend to revolve around signature analysis [McA87] [Wai87] [Sav88] [Wai89]. In addition to the straightforward signature by simulation scheme, the efforts are divided into two methods *intermediate signature collection* and *algebraic analysis.*

4.1 Signature by Simulation

The brute force scheme for fault location with data compaction is to construct a fault dictionary containing the signature of each modelled fault. Fault location using such a dictionary is straightforward, and aliasing behaviour can be determined directly The method has several drawbacks, however.

First, some physically distinct faults with different output behaviour may share the same signature — the chances of this occurrence depend on the compaction scheme used — and checking for this equivalence involves a sort at every proposed test length and ultimately guesswork in deciding when faults are equivalent.

Second, if an unmodelled fault occurs, this method can only report that no cause could be determined. No information as to the nature of the fault can be obtained Since dictionaries including all possible multiple faults are generally too large to be created (see section 2.3.3), this approach cannot hope to locate multiple failures

Finally, there is the amount of fault simulation required. The construction of a signature dictionary involves a full circuit simulation for each fault with every input pattern. Since test lengths when pseudo-random input vectors are employed tend to be longer than those in deterministic sets, such a construction could prove immensely expensive. Furthermore, dictionary construction must be performed after the design is finalized and before any diagnosis is attempted, which could place it on the critical path for circuit production. The time and expense could void the cost savings which justified the RCT or BIST in the first place

A demand-driven approach of constructing the dictionary as it was needed might appear to be a method of avoiding the critical path problem In such a scheme, simulation would be performed until a fault which resulted in a signature match was found Without any information about the fault which occurred, and if each fault has a unique signature, for the first diagnosis (assuming a modelled fault occurs) about half the total number of faults will have to be simulated on average before a match is found still a daunting task, and still potentially in the critical path.

The failings of the signature by simulation method result from the fact that the value of the signature is largely ignored. The signature contains information about the faulty output sequence, information which is potentially valuable in fault location. Not using it makes fault location more difficult and time consuming than it need be.

Potential solutions to the immense amount of effort required to construct a full dictionary in advance include simulation over a smaller test set, and simulating for a smaller fault set. Intermediate signature collection is primarily concerned with the former.

4.2 Intermediate Signature Collection

In intermediate signature collection (ISC) [Wai87] [Wai89], MISR signatures are taken every L patterns ($L = 256$ in [Wai87] [Wai89]) * During the test, failing signatures are noted (the MISR is reset after every L patterns, so errors are isolated to within a L pattern block), and the entire output sequence of L vectors for a failing block is stored. According to [Wai89], only a "few" of these sequences needs to be stored. Only those faults which could have caused the observed behaviour are retained in the list of potential faults, and these are simulated on the set of L patterns within each failing block. Once the test is complete, a revised list of potential faults is available. (Further simulation is presumably avoided if the fault list is reduced to a single fault). The scheme is shown in figure 4 1.

Waicukauski et al [Wai87] [Wai89] suggest a variety of methods to cut down on the size of the potential fault list, and hence the amount of simulation to be performed:

— There must be a path from the fault to the output on which it was observed.

— The fault path to the output must have correct parity for the failing pattern. This will eliminate one of the stuck-at 1 and stuck-at 0 faults at a given fault site, provided that no reconvergent fanout paths of different parity exist.

— The fault must be detectable at its fault site by the input pattern with which the error was observed.

— If a failure was observed on more than one output, the faults considered must satisfy the above constraints on each failing output

All of the above heuristics require exact circuit output information, hence the output storage area in figure 4.1. Further, some of them seem to resemble parallel fault simulation, in that path tracing in the presence of reconvergent fanout is not an easy

* The MISR used by the method is loaded in parallel at the primary outputs, then, for scan circuits, loaded serially as an LFSR with the scan outputs

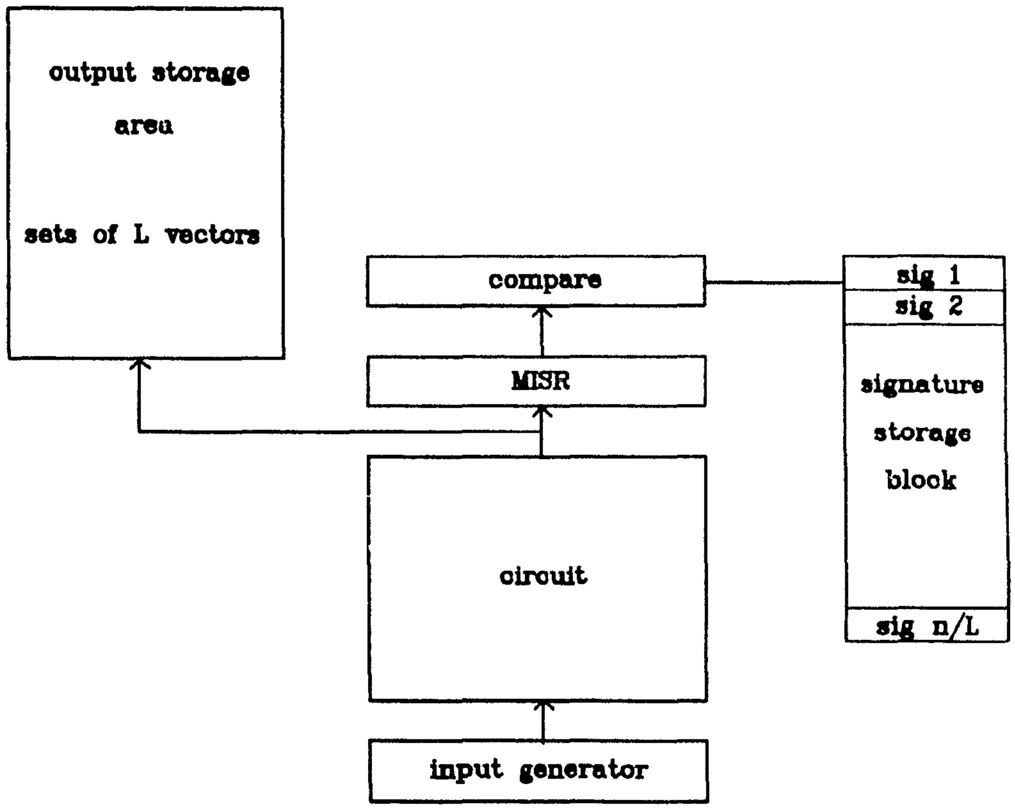


Figure 4.1 Intermediate Signature Collection

task. Waicukauski et al suggest using critical path tracing rather than parallel fault simulation. As a result, their heuristics may be of limited use in eliminating faults on fanout stems from consideration.

The authors assume three fault models. The first is a standard single stuck-at model. Such failures must account for every failing pattern and every passing pattern in their simulation (which simulates only single stuck-at faults) The second model is single-site non-stuck-at faults. These faults are assumed to behave as single stuck-at faults at the affected site for failing patterns, but not necessarily for passing ones This model can locate single transition faults and intermittent single stuck-at faults The final model is a limited multiple fault model This assumes that no circuit output is affected by more than one fault on any given test pattern The validity of this model is questionable, but it should explain more failures than either of the first two

The requirement for storage of actual output areas means that the ISC scheme cannot be entirely located on-chip. Thus, the scheme does not directly solve the problem of locating faults in circuits with BIST. Rather, it is a means of changing the problem of fault location in circuits with random compact testing to one of fault location with known faulty output sequences In this context, the scheme employs well-known principles Again, the use of a M[illegible] identify failing regions is almost irrelevant, since any signature could have [illegible] Thus, any information present in the MISR signature, besides the obvi[illegible] that the signature is not the expected value, is ignored.

The pot[illegible] gnostic resolution of intermediate signature collection is good, provided th[illegible] tiple failing blocks are observed. (The one experiment reported in [Wai87] stat[illegible] that failures were always resolvable to the single fault injected into the circuit). However, the overhead required by the method is large: storing the fault-free MISR signatures requires $L^{-1}m'n$ bits of information for an n vector test on a circuit with m' primary (not scan) outputs, and each set of L output vectors stored requires another mL bits, where m is the total number of outputs in scan mode ($m > m'$). The amount of simulation required at each step depends on the ability of the heuristics to eliminate faults. In the worst case, all faults must be simulated over at least one interval for each diagnosis, but this is unlikely in practice. The improvement over the worst-case will depend to a large extent on the amount of reconvergent fanout in the circuit The resolution of the method and the number of blocks to be simulated will be related to the ability to distinguish equivalent faults, since not all such faults can be determined through trivial identification schemes (e.g. the input and output of an AND gate stuck-at 0 are equivalent faults) In addition, the parity analysis and path tracing algorithms must be less complex than fault simulation if they are to be of any value.

It is interesting that ISC as proposed in [Wai87] and [Wai89] does not advocate retaining results in a fault dictionary Instead, a fraction of a dictionary is constructed at each diagnosis and then discarded. It is not stated directly whether this practice is because the storage requirements of a dictionary are too large or whether the effort required for individual diagnoses is viewed as acceptable without a dictionary, although the restricted multiple fault model implies the former.

Unfortunately, neither [Wai87] nor [Wai89] gives any examples of experiments showing the performance of ISC on any benchmark circuits, such as those of [Brg85], or an indication of the performance improvements gained by the heuristics Without these heuristics, only the fault-free signatures would have to be stored, and an on-chip implementation could be possible The performance of ISC under these constraints has been evaluated, and the results are given in chapter 9

A final drawback to intermediate signature collection, as described in [Wai87] and [Wai89], is that it cannot be applied to non-scan design sequential circuits, since these circuits cannot have their internal states reliably reset to the fault-free value every L

patterns. For this reason, only combinational circuits and full-scan sequential circuits are amenable to ISC.

4.3 Algebraic Analysis

This section describes a signature-based fault location method which attempts to locate a failing pattern before performing any simulation. In algebraic analysis [McA87] [Sav88] the dictionary size can potentially remain small, since there are either one [McA87] or a few [Sav88], signatures per fault. The method proposes to determine error patterns which could cause a given signature, then equate these to a fault. Resolution to a single fault may not be possible, since several faults may have the same signature. In essence, the method proposes to run an LFSR in reverse in order to arrive at a set of possible failures.

The first paper on algebraic analysis, [McA87], suggests using an LFSR with a primitive feedback polynomial. The LFSR should have a state space greater than the test length; that is, for an n vector test, the LFSR length, say j, should be greater than $\lfloor \log_2 n \rfloor$. McAnney and Savir then suggest computing a signature for each of the n possible single-bit errors. With the previous restriction on the LFSR, each of these errors will produce a different signature. The observed signature is then assumed to be a single-bit error and faults are simulated to see which are detected by the assumed erroneous pattern. If a unique diagnosis cannot be made, the process essentially degenerates into the signature by simulation method of section 4.1. Further, if a unique diagnosis is made, there is no guarantee that it is correct, since the signature may have actually resulted from several failing bits. Thus, for faults which cause more than a single bit error, algebraic analysis on an LFSR is difficult, time-consuming, and, for large number of errors, amounts to signature by simulation.

In a modified proposal [Sav88], Savir and McAnney propose using cycling registers [Dav80]. Through the use of 3 of these with relatively prime lengths, up to 50 errors in a sequence of 2^{20} bits can be diagnosed with reasonable accuracy using a signature with total length of about 300 bits. Note, though, that 300 bits per fault could result in a fairly large dictionary. Even so, locating a fault which causes hundreds or thousands of errors in the output stream remains infeasible with algebraic analysis, and cycling registers are believed to have much poorer aliasing characteristics than primitive LFSRs [Wil86]. In all cases, a single fault model is explicitly assumed, so locating faults in the presence of multiple faults is as infeasible as it is with the signature by simulation technique. Finally, the technique has so far been proposed only for single output circuits.

Each additional circuit output compounds the problem, so analysis of MISRs seems virtually impossible using the algebraic analysis techniques proposed so far.

4.4 Requirements for Fault Location in the RCT Environment

It appears that the greatest advantage of signature analyzers for data compaction is their greatest disadvantage in fault location: The relationship between faults and signatures is essentially random.

The discussions of the previous sections have shown that fault location when data compaction is used revolves around simulation. The reason for this is that the compacted output, the signature, contains little, if any, directly usable location information. The Algebraic Analysis method [McA87] [Sav88] is not applicable to multiple-output circuits, and its performance degrades as the number of errors increases The Intermediate Signature Collection method [Wai87] uses signatures only as indicators of blocks of failing patterns — the signature value itself does not aid in the location process.

A solution to the problem of fault location in the RCT environment must possess the following features:

1) Ability to extract location information from its signature(s).
2) Small fault dictionary size.
3) Resolution to a single fault equivalence class.
4) Potential for demand-driven dictionary construction.
5) Applicability to multiple-output combinational circuits.
6) Ability to use pseudo-random input vectors.
7) No requirement to store actual output vectors.
8) Potential hardware implementation.

The reasons for these goals are as follows:

1) When the signature contains information useful for fault location, a hierarchical approach becomes possible. Fault location with this approach becomes a series of reductions on the original fault set, which allows faults to be characterized by their output behaviour. In addition to this and other inherent benefits of hierarchical diagnosis, there is the esthetic benefit of not "wasting" the information present in the signature

2) If an entry to the fault dictionary is too large, there will be little justification in retaining the values once a diagnosis has been made This will result in unnecessary repetition of effort if the same fault recurs.

3) A scheme cannot do better than to resolve faults to a single equivalence class (such a class includes any faults with identical behaviour over the test sequence). Resolution

cannot be improved beyond this, since the number of faults within these classes is a function of the fault model and the test set, not the location scheme Nonetheless, in most cases, faults within these sets will be equivalent faults This level of resolution, termed "single-fault resolution," is the goal of all fault location methods.

4) If the fault dictionary must be constructed in advance, the critical path problem of its generation occurs. Generating a complete dictionary is a horrendous problem for large circuits tested with conventional methods, as simulation tools will spend inordinate amounts of time finding signatures for thousands of faults when each requires a simulation over hundreds of thousands of vectors.

5) This is a minimum requirement. This will permit fault location on full-scan design sequential circuits and purely combinational circuits. A method which would work on all multiple-output sequential circuits would be more general and hence more useful.

6) The use of (possibly weighted) pseudo-random input vectors permits a straightforward test pattern generator which may even be included on-chip. Exhaustive or pseudo-exhaustive sequences are too long for many circuits, and deterministic vectors cannot be stored on-chip.

7) When output vectors need not be stored, the overhead requirements of the method are greatly reduced, again allowing the possibility of on-chip implementation. Further, data compaction is used to reduce the amount of output data Storing output vectors, especially those which are error-free, is contrary to this end.

8) Potential hardware implementation follows from 6 and 7, and allows development of diagnosis methods as natural extensions of BIST techniques

The methods outlined in sections 4.1 through 4.3 do not meet criteria 1 through 8. Signature by simulation does not meet criteria 1 and 4. Intermediate signature collection, as proposed by [Wai87] and [Wai89], fails criteria 1, 2, 7, and 8, although it could conceivably be adapted to meet 7 and 8. Finally, algebraic analysis fails criteria 3 and 5, as well as criterion 2 for the implementation suggested in [Sav88]

With these goals in mind, a new fault location method is proposed in the next chapter. This method meets all 8 goals, and is the subject of the remainder of this dissertation.

Chapter 5 Hierarchical Fault Diagnosis: Theory and Background

This chapter proposes and outlines the benefits of a hierarchical method as a solution to the problem of fault location in testing schemes which use data compaction. This method, which is the subject of the remainder of this dissertation, uses information contained within the signature as a reliable aid to fault location. Such an approach treats diagnosis as a sequence of steps, each of which provides some information about which, if any, of the modelled faults may be present in a given circuit under observation. This differs from the conventional approach, where diagnosis is considered to be a one-step global dictionary search. It is shown that circuit behaviour can be used to characterize faults and eliminate some modelled faults from consideration because of inconsistencies between observed and predicted behaviour. As an additional benefit, the average amount of work performed can be reduced over that of conventional global techniques.

Potential benefits of a hierarchical approach include the ability to construct a fault dictionary in a demand-driven fashion, and thus amortize the start-up cost of dictionary construction over a product's lifespan, or alternately, use dictionary methods of diagnosis even when a complete dictionary is too large to construct or store. Other advantages are a diagnosis method whose resolution can be tailored to a user's needs, partial characterization of unmodelled faults, and the ability to evaluate fault models. Finally, there is the esthetic benefit of using available information about circuits as part of their diagnosis.

The constraint with the hierarchical approach to fault location with data compaction is identical to that with the all-or-nothing approach. Fault location is to be performed after the test is completed, so the actual output sequence is no longer available, only its compacted form as a signature. The choice of signature is clearly important. In addition, some form of fault simulation is also almost certainly required Since this simulation is expensive (see section 2.5.6.1), it must not be used indiscriminately. In the ideal case, its use would not be required at all.

The chapter is ordered as follows: First, an overview of the hierarchical method of fault diagnosis is given; second, the theoretical foundations of this hierarchical method are provided; and finally, the complexities of both determining diagnosis parameters from faulty circuits and calculating parameters from modelled faults are analyzed.

5.1 Outline of Proposed Technique

This section outlines the basic properties of the hierarchical fault diagnosis method proposed in this dissertation.

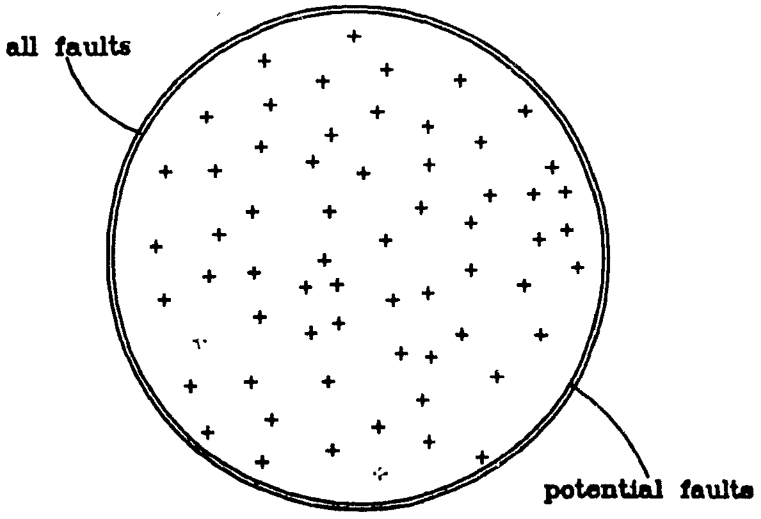


Figure 5.1 Potential Faults, Pass/Fail Signature

Consider a circuit which has produced a faulty signature during its test If the signature is a mere binary indicator (pass/fail), then there is no information available about which fault actually occurred The situation is as depicted in figure 5 1 On average, about half of the total number of faults will have to be investigated before the fault responsible for the erroneous signature is located

Suppose, on the other hand, that the signature was not simply a binary indicator, but also gave some idea as to a class of faulty circuits which was responsible With a suitably chosen classification scheme, many faults could be immediately eliminated from consideration without simulation. In other words, the signature would provide a

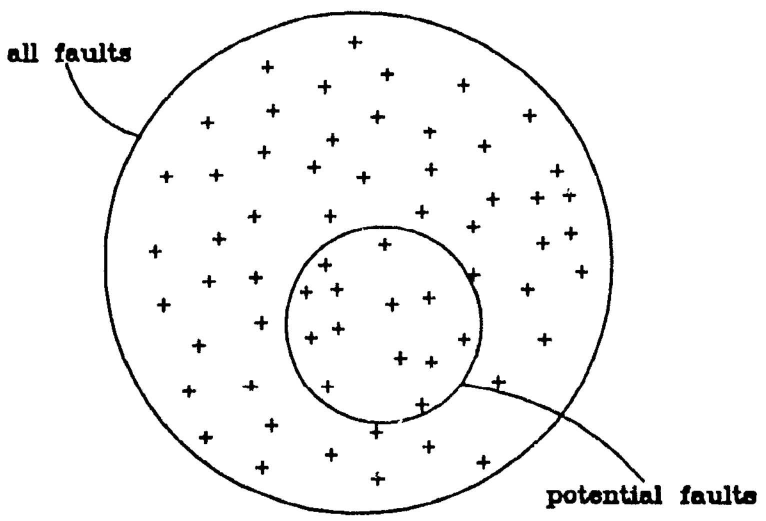


Figure 5.2 Potential Faults, After Coarse Resolution

form of coarse resolution for the faults, as shown in figure 5.2. The classification scheme chosen must clearly require simpler calculations than a full fault simulation. An example of such a scheme is random pattern detectability, as discussed in section 2.5.3. So, a signature which was able to indicate a property such as the random pattern detectability of the fault which occurred would provide coarse resolution to a fault location scheme.

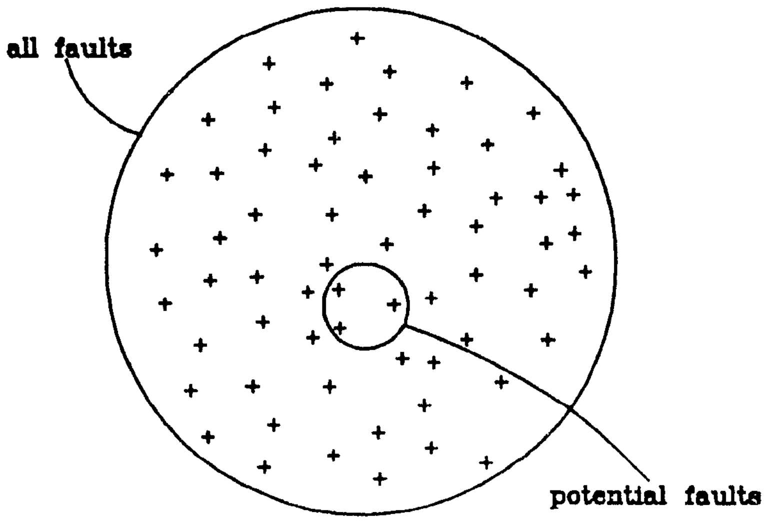


Figure 5.3 Potential Faults, After Further Reduction

Nonetheless, many faults might have similar detection probabilities, so some means would still be necessary to distinguish between these. Suppose that an additional circuit parameter was available from the signature. For example, in addition to detection probability, the number of the first input pattern which resulted in an error, the *first-fail*, might also be available. In this case, simulation to the full test length would not have to be performed to compare a modelled fault to the observed one whenever a discrepancy in the first-fail values occurred (simulation can stop after one has failed and the other has not). The result would be a further reduction in the number of potential faults. This situation is shown in figure 5 3.

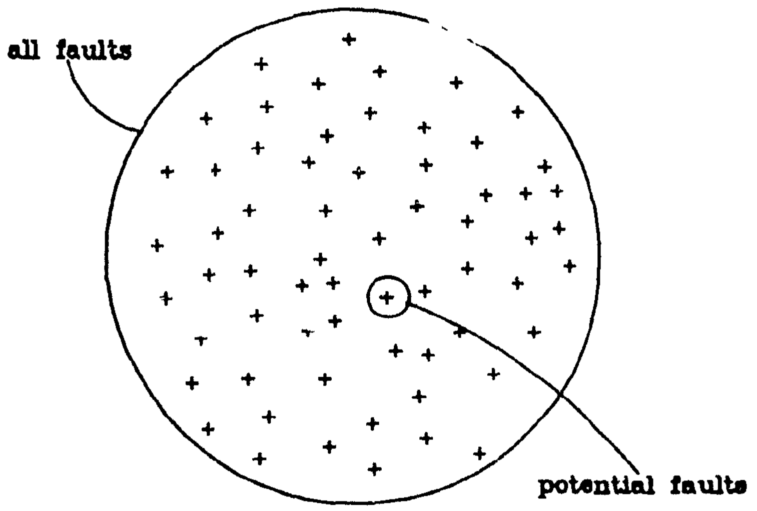


Figure 5.4 Potential Faults, Final Diagnosis

Additional parameters could continue to be added until the remaining fault set was sufficiently small. At this point, a final parameter, such as a conventional LFSR signature, could be added to provide fine resolution to the scheme Full simulation would only be performed on a small fraction of the total number of faults, and single-fault resolution would result, as shown in figure 5.4.

This is the hierarchical method proposed in this dissertation. The method is intended for circuits tested with random or pseudo-random test vectors. Some advantages of the method are·

- Fault classes can be developed which characterize faults by particular attributes of their output behaviour
- Fault dictionaries can be constructed in a demand-driven fashion, removing the dictionary construction process from the critical path in circuit production
- The logistics and complexities of searching a full dictionary can be avoided

- If storage is a problem, a complete fault dictionary need never be constructed. Instead, the necessary parts can be generated with every diagnosis.
- If no unique diagnosis can be made, or if an observed fault is not accounted for by the chosen fault model, some characterization of the fault is still possible.

Numerous implementations of the scheme are possible, depending on which attributes are selected for observation. It is shown that those mentioned above, detection probability and first-fail, are just two of the possibilities. Using the hierarchical approach, fault location becomes feasible with signature-based testing schemes.

5.2 Theoretical Analysis

This section provides theoretical justification for using a variety of circuit observations in order to improve the demand-driven construction of a fault dictionary. It expands formally on the advantages of the hierarchical method introduced last section.

5.2.1 Preliminary Definitions

To begin with, formal definitions are provided for some terms in fault diagnosis. Let B^k represent the set of binary vectors of length k; that is:

$$B^k \equiv \{\langle b_1, b_2, ..., b_k\rangle\}$$

where

$$b_i \in \{0,1\} \ \forall \ i; \ 1 \leq i \leq k$$

The set B^k is isomorphic to the set of all integers between 0 and $2^k - 1$ inclusive. This set is now used to define some terms needed in analyzing the performance of fault diagnosis methods.

Definition 5.2.1: The *output function*, G, implemented by a given combinational circuit Λ is defined as a mapping from B^l to B^m, where l is the number of circuit inputs and m is the number of circuit outputs.

Thus, G maps every input vector $v \in B^l$ is to a corresponding output vector $o \in B^m$.

Definition 5.2.2: A *test sequence*, V, for Λ consists of a sequence of input vectors $\langle v_1, v_2, ..., v_n\rangle$ where each $v_i \in B^l, 1 \leq i \leq n$

Definition 5.2.2a: A *random (pseudo-random) test sequence* for Λ is a test sequence where each vector is selected at random from B^l with (without) replacement.

Definition 5.2.3: The *expected output response*, O, for a given test sequence V and output function G is defined as $\langle o_1, o_2, ..., o_n\rangle$ where $o_i \equiv G(v_i), 1 \leq i \leq n$.

Definition 5.2.4: The *fault set* of Λ, F_Λ is defined to be a set $\{f_1, f_2, \ldots, f_h\}$. Associated with each fault $f_i \in F_\Lambda$ and test sequence V is a *faulty output function* ϕ_i which is a fixed mapping from V to B^m. The set of all such faulty output sequences forms the *faulty output set*, $\Phi_{\Lambda,V}$.

Definition 5.2.4a: A *fault sequence* is a fixed ordering of a fault set. Similarly, a *faulty output sequence* is a fixed ordering of a faulty output set.

In most cases it will be clear from context as to whether a given object is being referred to as a fault sequence or a fault set, hence the same symbol will often be used for both.

Each element of $\Phi_{\Lambda,V}$ represents the output function which results from a particular physical failure within Λ when it is tested with vector set V. Any permanent, deterministic circuit failures, including stuck-at, stuck-open, transition, and bridging faults may be represented using this notation.

The concept of the fault set is now used to define certain attributes and types of faults (In the following, $\Phi_{\Lambda,V} = \{\phi_i\}$, $1 \leq i \leq h$ is a faulty output set of length h whose elements correspond to faults in the set F_Λ when tested with V, a test sequence of length n. O is the expected response from G, the fault-free output function of Λ):

Definition 5.2.5: A given fault $f_i, 1 \leq i \leq h$ is *detected* by a test sequence V with expected output response O if and only if (iff) there exists j, $1 \leq j \leq n$ such that $\phi_i(v_j) \neq o_j$.

Definition 5.2.5a: The *first failing pattern*, r, for fault f_i is defined as the smallest j, $1 \leq j \leq n$ such that $\phi_i(v_j) \neq o_j$. If f_i is not detected by V, then r_i is defined to be $n + 1$. r_i may also be written $r_i^{(1)}$.

Definition 5.2.5b: The *kth failing pattern*, $r^{(k)}, k > 1$, for fault f_i is defined as the smallest j, $r_i^{(k-1)} < j \leq n$ such that $\phi_i(v_j) \neq o_j$. If no such j exists, then $r_i^{(k)}$ is defined to be $n + 1$.

Definition 5.2.5c: The *last failing pattern*, $r^{(\infty)}$, for fault f_i is defined as the largest j, $1 \leq j \leq n$ such that $\phi_i(v_j) \neq o_j$. If f_i is not detected by V, then $r_i^{(\infty)}$ is defined to be 0.

Definition 5.2.5d: The *failing output set*, $Z = z_1, z_2, \ldots, z_m$, is defined for faulty output function ϕ_i such that z_j is 1 if there exists k such that $\phi_i(v_k)$ differs from o_k at bit j, and z_j is 0 otherwise.

Definition 5.2.6: A given fault $f_i, 1 \leq i \leq h$ is *detectable* iff there exists a test sequence V and a vector $v \in V$ such that $\phi_i(v) \neq G(v)$.

Definition 5.2.7: A fault is *redundant* iff it is not detectable

Definition 5.2.8: Two faults f_i and f_j, $1 \leq i, j \leq h, i \neq j$, are ***equivalent*** iff there does not exist test sequence V' such that $\phi_i(v) \neq \phi_j(v)$ for some $v \in V'$ Faults f_i and f_j are ***equivalent over test sequence*** V iff $\phi_i(v) = \phi_j(v)\ \forall\ v \in V$.

Lemma 5.2.1: Two faults f_i and f_j which are equivalent over test sequence V cannot be differentiated by examining the effect of their output functions ϕ_i and ϕ_j over vectors in V.

Proof: Follows from definition.

Definition 5.2.9: The ***irredundant fault coverage***, C, of test sequence V with respect to fault set F_Λ of size h is given by

$$C \equiv \frac{\sum_{i=1}^{h} s_i}{\sum_{i=1}^{h} t_i}$$

where

$$s_i = \begin{cases} 1, & \text{if fault } f_i \text{ is detected by } V; \\ 0, & \text{otherwise.} \end{cases}$$

and

$$t_i = \begin{cases} 1, & \text{if fault } f_i \text{ is detectable;} \\ 0, & \text{otherwise.} \end{cases}$$

Definition 5.2.10: A test sequence is ***complete*** when its irredundant fault coverage is 1.*

Notice that a test sequence can only be complete with respect to a fault set, which is in turn derived from a fault model. The definitions above are now used to examine the behaviour of fault diagnosis methods.

5.2.2 Classes of Hierarchical Observations

There are two ways in which observed behaviours can be used to hierarchically reduce the set of potential faults. In the first, a fault sequence can be ranked as a result of some observation, so that the more likely a given modelled fault is to be the cause of the observed behaviour, the earlier it appears in the sequence. In the second, a predicted fault will possess a known value for particular attribute Faults can be grouped by attribute and value, allowing easy identification of potential faults when the attribute's value is observed on an actual circuit. These techniques, known as *fault ranking* and *fault partitioning* respectively, are now explained more formally

* Most of the fault diagnosis literature gives coverage values in percentages, so a complete test sequence has 100% irredundant fault coverage The measure is referred to as *irredundant* fault coverage, since fault coverage on its own is typically expressed as the ratio of detected faults to modelled faults

Definition 5.2.11: A *fault ranking step*, Ψ, for fault sequence F is a ranking, or reordering, of F.

Definition 5.2.11a: A *random fault ranking step*, Ψ_r, for fault sequence F is a ranking of F randomly selected from the set of all possible rankings

Definition 5.2.11b: A *probabilistic fault ranking step*, Ψ_X, for fault sequence F and observation X is a ranking of F subject to the probability of each fault given an observation X.

A probabilistic fault ranking step can be thought of as an ordering of faults from the most likely to least likely to have caused a particular observed behaviour. For example, if some number of errors has been counted at a faulty circuit output, fault detection probability at that output could be used in a fault ranking step, as shown below·

Definition 5.2.11c: An *expected count fault ranking*, $\Psi^{(E)}$ for fault sequence F of length h and output sequence X where w errors have been counted is defined as a ranking of F

$$\Psi_X^{(E)}(F) \equiv \langle \psi_1, \psi_2, ..., \psi_h \rangle$$

such that given ψ_i and ψ_j in $\Psi_X^{(E)}$, $i < j$ implies that $P(c_i = w) \geq P(c_j = w)$ where $P(c_k = w)$ refers to the probability that the count for the kth fault is w

This ranking is not guaranteed to be unique, but it is clear that one exists for every fixed w and F where probability values can be obtained.

Definition 5.2.12: The *ranking probability vector*, $q = \langle q_1, q_2, .., q_h \rangle$ associated with ranking Ψ and observation X is defined such that

$$q_i \equiv \frac{P(X \mid Y = f_i)}{P_t}$$

where Y is a randomly selected fault from F_Λ and

$$P_t \equiv \sum_{j=1}^{h} P(X \mid Y = f_j)$$

Each element q_i of the ranking probability vector is thus the relative probability of fault i given observation X. The sum of all the q_is is 1

Definition 5.2.13: A *fault partitioning step*, Γ, for fault set F is a partitioning of F into disjoint subsets $F_1, F_2, \ldots, F_N$

To be useful in practice, a fault partitioning step should group faults based on a fixed and predictable behaviour. This will allow the subset containing an observed fault to be identified immediately. An example of such a criterion for a fault partitioning step is the first failing pattern in a test sequence

5.2.3 Diagnostic Performance of Hierarchical Observations

This section shows how fault diagnosis using fault partitioning and fault ranking techniques can identify a fault with fewer comparisons than would be required otherwise.

Definition 5.2.14: A *fault identification step*, I, applied to two faults f_a and f_b will determine whether or not f_a and f_b are equivalent over test sequence V.

During fault diagnosis, a fault identification step will be used to determine whether or not a predicted fault precisely models an observed fault. It is assumed that the time required for a fault identification step between any two faults in a fault set will be roughly constant for a given test sequence V.

Definition 5.2.15: The *fault identification time* for fault f_i in fault sequence F is defined to be i, the position of f_i within F.

For a fault sequence F which contains no faults equivalent over test sequence V, the fault identification time can be thought of as the number of fault identification steps required, given an unknown fault $f_i \in F$, to locate, by simulation for example, f_i within F, assuming that location begins with the first fault in F. If each fault identification step is assumed to take a constant amount of time, the fault identification time gives an estimate of the average total effort required to identify the fault responsible for a given output sequence in the absence of any other information.

Definition 5.2.16: The *expected fault identification time*, t_{av}, for a fault sequence F is defined as:

$$t_{av}(F) \equiv \sum_{i=1}^{h} i \cdot P(Y = f_i)$$

where h is the size of F, and Y is a randomly selected fault from F.

Definition 5.2.16a: The expected fault identification time of a fault set F_Λ is defined to be the expected fault identification time of a random ordering of F_Λ.

Definition 5.2.16b: The expected fault identification time of a ranked fault sequence $\Psi_X(F)$ related to an observation X is defined as:

$$t_{av}(\Psi_X(F)) \equiv \sum_{i=1}^{h} i q_i$$

where $q = \langle q_1, q_2, \ldots, q_h \rangle$ is the ranking probability vector associated with Ψ_X and X, and h is the size of F.

In the case of a random fault ranking, X and Ψ_X are independent, so each $P(X \mid Y = f_i)$ is equal to $P(X)$, P_t is $hP(X)$, and each $q_i = \frac{1}{h}$.

Definition 5.2.16c: For a partitioned fault set $\Gamma(F)$, t_{av} is defined as the weighted sum of the individual partition values:

$$t_{av}(\Gamma(F)) = \sum_{i=1}^{N} \frac{|F_i|}{|F|} t_{av}(F_i)$$

where $\Gamma(F) = F_1, F_2, ..., F_N$ is a fault partition step

Lemma 5.2.2: The expected fault identification time of a random fault ranking step for fault sequence F is $\frac{h+1}{2}$ where h is the size of F and all faults are equiprobable.

Proof: Follows from definition.

Lemma 5.2.3: $t_{av}(\Gamma(F)) \leq t_{av}(F_{max})$ where $|F_{max}| \geq |F_i| \; \forall \; i, 1 \leq i \leq N$ where all faults in F are equiprobable.

Proof: Follows from definition.

Lemma 5.2.3a: $t_{av}(\Gamma(F)) \leq t_{max}$ where $t_{max} \geq t_{av}(F_i) \; \forall \; i, 1 \leq i \leq N$

Proof: Follows from definition.

Lemma 5.2.3a states that the expected fault identification time of a partitioned fault sequence is less than or equal to the largest fault identification time of any subsequence

Definition 5.2.17: A fault ranking step or a fault partition step is called a *predictor* for a fault sequence F with size h if its expected fault identification time is less than $\frac{h+1}{2}$. A sequence of such steps is called a *predicting sequence* if its expected fault identification time is less than $\frac{h+1}{2}$

Definition 5.2.18: The *speed-up* of a sequence of fault ranking steps and fault partition steps applied to an initial fault sequence F of size h is defined to be the ratio of $\frac{h+1}{2}$ to the expected fault identification time A predicting sequence thus has speed-up > 1.

A predicting sequence permits fault location to take place with fewer fault identification steps than would be required if the predicting sequence were not used Such a sequence is useful because it allows the hierarchical reduction of the problem of fault location. Ranking steps Ψ and partitioning steps Γ should be simpler than the identification steps they replace, and any parameters they require should be readily obtainable from a given circuit output sequence, so that the total time taken for diagnosis using a predicting sequence will be lower than would be required without it

Several well-known attributes of faults can be shown to be predictors using the following theorem.

Theorem 5.2.4: Any fault partitioning step Γ which divides a fault sequence F of size h into at least 2 non-empty sets is a predictor, provided that all faults in F are equiprobable.

Proof: Let Γ partition F into $F_1, F_2, ..., F_N$. Assume, without loss of generality, that these are ordered such that $|F_1| \geq |F_2| \geq ... \geq |F_N|$ Since $|F_1|, |F_2| > 0$ and $|F_1| + |F_2| \leq h$, it is clear that $|F_1| < h$. By Lemma 5.2.3, $t_{av}(\Gamma(F)) \leq t_{av}(F_1)$, and by Lemma 5.2.2, $t_{av}(F_1) = \frac{|F_1|+1}{2}$ Thus, $t_{av}(\Gamma(F)) < \frac{h+1}{2}$ and Γ is a predictor

Theorem 5.2.4 states that any identifiable property of a randomly ordered fault sequence F_Λ is a potential predictor. Several examples are now given.

Corollary 5.2.4a: The first failing pattern r_a of faults $f_a \in F_\Lambda$ is a predictor provided that there exist i and j such that $r_i \neq r_j$.

Corollary 5.2.4b: The kth failing pattern $r_a^{(k)}$ of faults $f_a \in F_\Lambda$ is a predictor provided that there exist i and j such that $r_i^{(k)} \neq r_j^{(k)}$.

Corollary 5.2.4c: The last failing pattern $r_a^{(\infty)}$ of faults $f_a \in F_\Lambda$ is a predictor provided that there exist i and j such that $r_i^{(\infty)} \neq r_j^{(\infty)}$

Corollary 5.2.4d: The first (last) failing pattern is always a predictor for a test sequence which is complete with respect to output stuck-at faults.

Proof: Follows from the facts that no test vector can detect an output line stuck at both 0 and 1 and that both such faults must be detected by a complete test sequence.

Corollary 5.2.4e: The failing output set is always a predictor for a circuit of 2 or more outputs and a test sequence which is complete with respect to output stuck-at faults.

Corollary 5.2.4f: The detection probability p_a of faults $f_a \in F_\Lambda$ is a predictor provided that there exist i and j such that $p_i \neq p_j$. This will be true when one output has a signal probability different from 0.5 and the test sequence includes both stuck-at faults for that output (This condition is sufficient, but not necessary)

Clearly, corollary 5 2 4f will apply regardless of the manner in which detection probability is calculated: i.e., on individual outputs or combinations, although it assumes exact results. Methods of dealing with approximations will be analyzed later.

Corollary 5.2.4g: A signature $s(f_a)$ of faults $f_a \in F_\Lambda$ is a predictor provided that there exist i and j such that $s(f_i) \neq s(f_j)$

All of the signatures described in chapter 3 satisfy the requirements of corollary 5.2 4g for most circuits under commonly used fault models In addition, signature values for these circuits are readily available from fabricated circuits with fairly simple hardware However, the requirements for predictors also include complexity of calculation as compared with fault identification. Section 5 4 shows that this may be a problem

In addition to the predictors listed above, any other technique which is able to eliminate some faults from consideration without a fault identification step is a predictor. Methods such as the "effect-cause" analysis of [Abr80] fall into this category.

Rankings made independently of circuit information are in general not predictors. A specific ordering may be a predictor for a given circuit under test, but its predicting capabilities may not apply to the next unit observed. In addition, a fault partition where the fault is not isolated to a known partition is not a predictor. Most circuit attributes are predictors because they provide some location information, whereas a random search uses none. In addition to prediction capability, speed-up is an important attribute in practical implementations. Speed-ups should be measured in orders of magnitude, which will eliminate some potential predictors from practical consideration. For example, if a partitioning step divides the faults into 1000 groups of approximately equal size, a speed-up of at least 500 will be obtained on average. This speed-up makes the particular partition more attractive than one which divided the faults into only 10 equal-sized groups, even though both are predictors. Section 5.4 elaborates on this point.

By definition 5.2.16, the expected fault identification time of a ranked fault set is related to the probability of occurrence for each fault. If a ranking is made as the result of some observation such that the probability of faults early in the fault list tends to be higher than those later in the fault list, then the ranking will be a predictor. This concept is formalized in the following.

Definition 5.2.19: A *reduction perturbation*, $R(q, i, j, \delta)$, transforms a probability vector $q = \langle q_1, q_2, \ldots q_h \rangle$, into another probability vector q' such that

$$q' = \langle q_1 + \delta_a, q_2 + \delta_a, \ldots, q_i + \delta_a, q_{i+1}, q_{i+2}, \ldots, q_{j-1}, q_j - \delta_b, q_{j+1} - \delta_b, \ldots, q_h - \delta_b \rangle$$

for $i, j : 1 \leq i < j \leq h$ and where $\delta_a = \frac{\delta}{i}$, $\delta_b = \frac{\delta}{h-j+1}$, for fixed $\delta > 0$ such that $q_h - \delta_b \geq 0$ and $q_1 + \delta_a \leq 1$.

Lemma 5.2.5: Given a ranking probability vector $q = \langle q_1, q_2, \ldots q_h \rangle$, a reduction perturbation $R(q, i, j, \delta)$ of q implies that $t_{av}(q') < t_{av}(q)$.

Proof: Let

$$q' = \langle q_1 + \delta_a, q_2 + \delta_a, \ldots, q_i + \delta_a, q_{i+1}, q_{i+2}, \ldots, q_{j-1}, q_j - \delta_b, q_{j+1} - \delta_b, \ldots, q_h - \delta_b \rangle$$

for $i, j : 1 \leq i < j \leq h$ where $\delta_a = \frac{\delta}{i}$, $\delta_b = \frac{\delta}{h-j+1}$, for fixed $\delta > 0$, and let $t = t_{av}(q') - t_{av}(q)$. It is necessary to show that the perturbation implies that $t < 0$. From the definition of t_{av}, it follows that

$$t = \delta_a \sum_{k=1}^{i} k - \delta_b \sum_{k=j}^{h} k$$

which reduces to

$$t = \frac{\delta}{2}\left(i + 1 - \frac{1}{h - j + 1}(h(h+1) - j(j-1))\right)$$

To show that $t < 0$, it is thus necessary to show that

$$(i+1)(h - j + 1) < h(h+1) - j(j-1)$$

or

$$h^2 - j^2 + 2j > ih - ij + i + 1$$

If $j = h$, the above is obvious by inspection, since $i < j$ which implies that $h > 1$. If $j < h$, then the previous inequality may be demonstrated by showing

$$(h+j)(h-j) > i(h-j) \quad \text{and} \quad 2j \geq i + 1$$

both of which are clear by inspection. QED

Definition 5.2.20: A *well-formed ranking* is a ranking Ψ where the probability vector q associated with Ψ and some observation X is such that $q_1 \geq q_2 \geq \ldots \geq q_h$ and $q_1 > q_h$.

Lemma 5.2.6: The expected count ranking is a well-formed ranking.

Proof: Follows from definition.

Lemma 5.2.7: Every well-formed ranking Ψ can be formed by a sequence of reduction perturbations from an equal probability vector

Proof: Let q be the probability vector associated with Ψ and X, and let $\overline{q}$ be the equal probability vector $\langle\frac{1}{h}, \ldots, \frac{1}{h}\rangle$ Define Δq such that $\Delta q_k = q_k - \overline{q}_k$ and choose i and j such that Δq_i is the smallest positive element of Δq and Δq_j is the smallest negative element of Δq. Clearly, $i < j$. If $\Delta q_i < -\Delta q_j$, then apply reduction ranking $R(q, i, j, i\Delta q_i)$ giving vector q'. The elements of q' are as follows:

$$q'_k = \begin{cases} q_i & \text{for } k \leq i, \\ q_k & \text{for } i < k < j, \\ q_j + \delta_h & \text{for } k \geq j. \end{cases}$$

where $\delta_l = \frac{i\Delta q_i}{h - j + 1}$ Similarly, if $\Delta q_i \geq -\Delta q_j$, apply reduction ranking $R(q, i, j, (h - j + 1)\Delta q_j)$, giving q' whose elements are

$$q'_k = \begin{cases} q_i + \delta_a & \text{for } k \leq i, \\ q_k & \text{for } i < k < j, \\ q_j & \text{for } k \geq j, \end{cases}$$

where $\delta_a = \frac{-(h-j+1)\Delta q_j}{i}$. In both cases, q' is well-formed. The process is continued, letting $\Delta q = q - q'$ until $q = q'$, which will occur in less than h steps QED

Theorem 5.2.8: A well-formed ranking is a predictor

Proof: By Lemma 5.2.7, every well-formed ranking Ψ can be formed by a sequence of reduction perturbations from an equal probability vector By Lemma 5.2.5, every such perturbation reduces t_{av}, and since a predictor has t_{av} smaller than an equal probability vector it follows that every well-formed ranking is a predictor

Corollary 5.2.8: The expected count ranking is a predictor

In fact, the previous theorem is restrictive. A well-formed ranking is sufficient as a predictor, but far from necessary, as shown by the next theorem

Theorem 5.2.9: If and only if the correlation between the elements of a probability vector $q = \langle q_1, q_2, \ldots, q_h \rangle$ and its index elements is negative, then the ranking Ψ represented by q is a predictor

Proof: Correlation ρ between X and Y is defined as.

$$\rho = \frac{1}{h-1} \sum_{i=1}^{h} (X_i - E(X))(Y_i - E(Y))$$

which in this case is:

$$\rho = \frac{1}{h-1} \sum_{i=1}^{h} \left(q_i - \frac{1}{h}\right)\left(i - \frac{h+1}{2}\right)$$

since $E(q_i) = \frac{1}{h}$ and $E(i) = \frac{h+1}{2}$. Because $h > 1$, only the summation $\sigma = \rho(h-1)$ needs to be considered in determining whether or not $\rho < 0$. Expanding this summation yields

$$\sigma = \sum_{i=1}^{h} \left(q_i i - \frac{i}{h} + \frac{h+1}{2h} - q_i \frac{h+1}{2}\right)$$

Since $\sum_{i=1}^{h} q_i = 1$, the above reduces to

$$\sigma = \sum_{i=1}^{h} q_i i - \frac{h+1}{2} + \frac{h+1}{2} - \frac{h+1}{2}$$

$$= \sum_{i=1}^{h} q_i i - \frac{h+1}{2}$$

Since every $q_i \geq 0$, $\sigma < 0$ if and only if

$$\sum_{i=1}^{h} q_i i < \frac{h+1}{2}$$

which is precisely the definition of a predictor QED

5.3 Constraints on Parameter Observation

So far, it has been shown that detection probability may be used as a predictor both directly, by partitioning the fault set, and indirectly, through the expected count ranking. While a partitioning step is more desirable, it is now shown that it is likely to be infeasible in practice. The following lemmas show that exact values cannot be obtained by observing a circuit's output sequence for non-exhaustive tests.

Lemma 5.3.1: Given a description of circuit Λ, calculation of exact detection probability is a $\#$P-Complete problem in the number of inputs to Λ.

Proof: See [Kri86]

Lemma 5.3.2: Let a, b, c, d and e be fixed positive constants. Given a polynomial-sized circuit Λ with l inputs ($|\Lambda|$ is $O(l^a)$), if exact detection probability values can be obtained by any polynomial-time operation ($O(l^b)$) on the output sequence X of a possibly faulty circuit Λ when the length of test sequence V is of polynomial order (i.e., the length of the test sequence, n, is $O(l^c)$) then P $=$ $\#$P, where P and $\#$P are as defined in [Gar78]

Proof: An output sequence X for a given fault can be produced in polynomial time with respect to n and $|\Lambda|$ ($O(n^d|\Lambda|^e)$), and hence l (since n is $O(l^c)$ and Λ is $O(l^a)$), by simulation. If exact detection probability could be obtained by some operation of complexity $O(l^b)$ on X, then the entire complexity of calculating exact detection probability would be $O(l^{abcde})$. This operation is polynomial in l, which would show that $\#$P $=$ P by Lemma 5.3.1. Such a conjecture is generally regarded as highly unlikely [Gar78]. QED

While exact calculation of detection probability from a given output sequence remains a difficult and open problem, its value can be readily estimated from a given error sequence, using the ratio of errors to sequence length. Estimated detection probability values could then be used to create an expected count ranking. However, this ranking has been shown to be well-formed only when accurate detection probability values are used. When only an estimate is available, the results could be affected somewhat.

Using inaccurate values for detection probability to produce an expected count ranking could result in some faults being placed into the fault list after others with a lower probability of causing a given observation. This will not affect the ranking's ability as a predictor, provided that the negative correlation between actual probability values and their corresponding index within the fault list is maintained. Clearly, the greater the speed-up of the ranking (the further t_{av} is from $\frac{n+1}{2}$), the smaller the chance that inaccuracies in detection probability estimates will eliminate the predicting properties of the ranking.

While detection probabilities cannot be observed exactly, the other predictors mentioned in this section, including counts for approximate detection probability can all be readily observed from either the output function of the circuit under test, or its error function (output function exclusive-or'ed with its fault-free value). Potential predictors which involve circuit internals, such as the internal line or set of lines involved in a fault, may be difficult to observe directly from output functions, especially in the presence of reconvergent fanout. Methods such as those of [Abr80] and [Raj87], which show the *absence* of faults on given lines may be useful as predictors, depending on the complexity of their calculations, particularly in the case of multiple fault models.

5.4 Complexity of Predicting Observations

In section 5.2 it was shown that many circuit properties are potential predictors. The purpose of predictors is to hierarchically reduce the complexity of fault diagnosis by reducing the number of potential faults at each step of the diagnosis process. Using a predictor thus reduces the number of fault identification steps.

In order for this reduction to be useful, calculating the predicting parameter must be less complex than performing the fault identification steps otherwise required. First then, the complexity of fault identification is shown.

Lemma 5.4.1: Given a test sequence V, and faulty output functions ϕ_i and ϕ_j, the worst-case complexity of showing that ϕ_i and ϕ_j are equivalent over V is $\Omega(mn)$* where m is the number of circuit outputs and n is the length of V.

Proof: Every output function (faulty or fault-free) of V consists of n m-bit vectors. Any two such functions ϕ_i and ϕ_j can clearly be compared in mn steps. Now, ϕ_i and ϕ_j could differ by only a single bit. Since this bit cannot be known *a priori*, each of the mn steps may be required in the worst case to show equality, and the complexity result follows.

So, if a predicting parameter is to be useful, it should be possible to calculate it in fewer than $O(mn)$ steps. The parameters identified as predictors in the previous section are now analyzed.

The expected count permutation uses detection probability which may be estimated using the methods described in section 2.5.3. For reasonable values of m and n calculation of detection probability will be much simpler than fault identification.

* A function f is $\Omega(g(n))$ if for every h such that f is $O(h(n))$, g is $O(h(n))$.

Lemma 5.4.2: Given an output function, ϕ, the complexity of calculating a parameter s whose value can be affected by k bits of ϕ is $\Omega(k)$ when examining each bit operation is $\Omega(1)$

Proof: Follows directly.

Corollary 5.4.2a: Given ϕ_a, the complexity of calculating the first failing pattern r_a for fault $f_a \in F_\Lambda$ is $\Omega(mr_a)$.

Corollary 5.4.2b: Given ϕ_a, the complexity of calculating the kth failing pattern $r_a^{(k)}$ for fault $f_a \in F_\Lambda$ is $\Omega(mr_a^{(k)})$.

Corollary 5.4.2c: Given ϕ_a, the complexity of calculating the last failing pattern $r_a^{(\infty)}$ for fault $f_a \in F_\Lambda$ is $\Omega(m(n - r_a^{(\infty)}))$.

Corollary 5.4.2d: Given ϕ_a, the complexity of calculating the failing output set for fault $f_a \in F_\Lambda$ over test sequence V is $\Omega(mn)$.

Corollary 5.4.2e: Given ϕ_a, the complexity of calculating a signature which depends on all output bits for fault $f_a \in F_\Lambda$ over test sequence V is $\Omega(mn)$

These results are now examined in detail.

5.4.1 Failing Pattern Measurements

Corollaries 5.4.2a through 5.4.2c show that the complexity of calculating failing pattern measurements for given output functions is no worse than performing fault identification This section will show that on average it can be much less For simplicity, only the first failing pattern will be considered. It is demonstrated in chapter 8 that the other values tend to follow from it.

Identifying whether or not a pattern has failed involves a comparison with each of the fault-free output bits, for a total of m comparisons. Thus, m will be a common factor in all values and will be ignored. The focus will instead be on r and n, specifically to demonstrate that $r_{av} < n$

Consider the set of first failing patterns, $\{r_i\}$, each element of which corresponds to a particular fault $f_i \in F_\Lambda$. Let F be a sorted fault sequence such that given f_i and f_j in F, $i < j$ implies that $r_i \le r_j$ These can then be plotted as shown by the typical example in figure 5.5 Notice that this figure shows fault coverage as it varies with test length The area above the curve corresponds to the number of output bit comparisons required to calculate the first failing pattern for every fault, while the area of the entire graph corresponds to the number required for complete fault identification The relative areas also reflect the amount of fault simulation required in each case (see section 5.4.4 for more details) The average reduction is clearly visible.

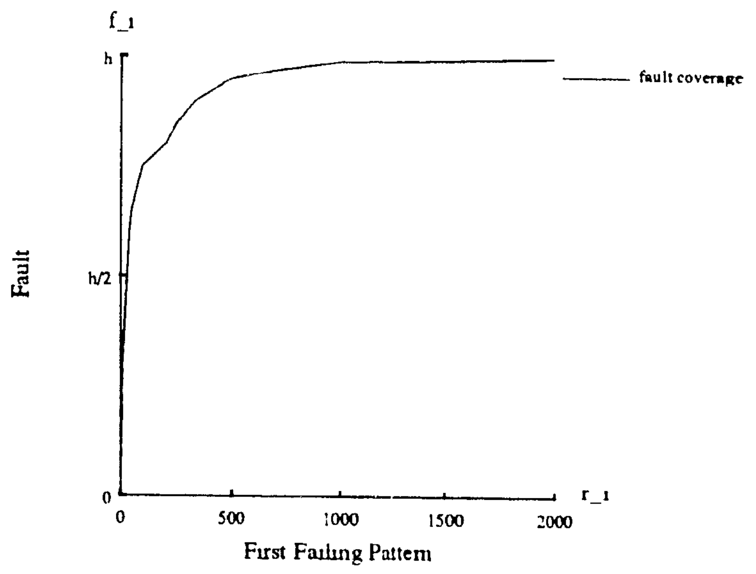


Figure 5.5 First Failing Pattern or Fault Coverage Curve

Several key observations may be made about fault coverage curves.

- The curve increases monotonically and asymptotically to 100% irredundant coverage.*
- The area above the curve is small in comparison with the area below it
- The ratio of area above the curve to area below the curve decreases as test length increases.

These observations indicate that the amount of fault simulation required to calculate the first failing pattern for all faults is a small fraction of the amount required to obtain a complete signature for all faults, and that the larger the test length, the smaller this fraction will be. This is now demonstrated by modelling the fault coverage curve.

The characteristic shape of the fault coverage curve is reminiscent of the arctangent function in terms of its asymptotic approach to irredundant fault coverage of 1 Because of this similarity, the following model is proposed·

$$C' = \frac{2}{\pi}(\tan^{-1}(\alpha\eta)^{\frac{1}{b}})$$

where η is test length, $\alpha > 0$ is a real constant and b is a positive integer No physical justification for this model is provided, since it is merely offered as a sample curve fit

* Redundant faults add a constant fraction to the total effort i e if 1% of all faults are redundant, a constant 1% of the total graph area will be added to the total first-failing pattern cost increasing r_{av} At some test length, this effort will be greater than that required by a deterministic redundancy identifier (e g [Sch88]) to eliminate the redundant faults from the fault list

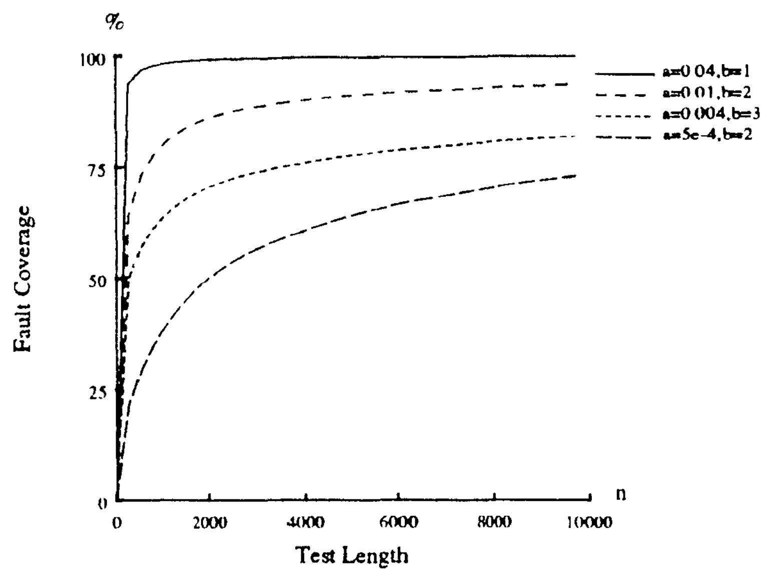


Figure 5.6 Arctangent Model of Fault Coverage

y for knee	b	α
> 0.95	1	$\frac{12}{x}$
$0.85 - 0.95$	2	$\frac{17}{x}$
$0.75 - 0.85$	3	$\frac{21}{x}$

Table 5.1 Determining coefficients for arctan model, knee at (x, y)

Instead, examples are given in figure 5.6 to show the variety of fault coverage curves which can be generated using the model.

Using the arctangent model, the value of r_{av} can be obtained as follows:

$$r_{av} = \frac{1}{\alpha}\int_0^{c_m} \tan\left(\frac{\pi}{2}c\right) dc + (1 - c_m)n$$

where n is the maximum test length and c_m is the fault coverage at n, so

$$c_m = \frac{2}{\pi}\tan^{-1}\left((\alpha n)^{\frac{1}{b}}\right)$$

A given fault coverage curve can be approximated by this model as shown in table 5.1. Given a knee at position (x, y), e.g. $(600, 0.95)$ in figure 5.5, the coefficients α and b can be determined from the appropriate columns. For figure 5.5, $\alpha = 0.02$ and $b = 1$. The curve and its approximation are shown in figure 5.7. Using nonintegral values for b permits closer fits, but makes analysis more difficult.

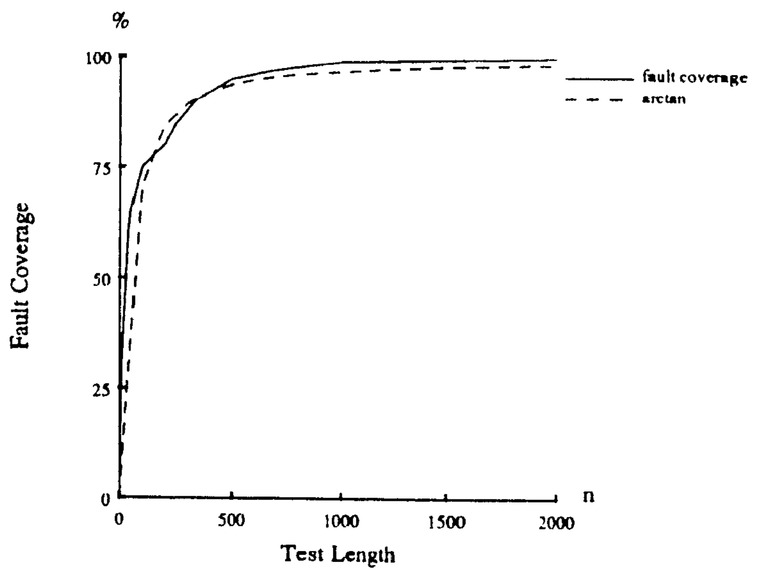


Figure 5.7 Fitting an arctangent curve to fault coverage

Evaluating the arctangent function for a few values of b yields the following.
Case 1: ($b = 1$, function is very random testable)

$$r_{av} \approx \frac{2}{\alpha\pi}\ln(\alpha n) + \left(1 - \frac{2}{\pi}\tan^{-1}(\alpha n)\right) n$$

As the coverage of the test sequence approaches 100%, the logarithmic term will dominate, showing that r_{av} is $O(\ln n)$ while fault identification is $O(n)$, assuming that redundancies are eliminated in advance.
Case 2: ($b = 2$, function is somewhat random testable)

$$r_{av} \approx \frac{2}{\alpha\pi}\sqrt{\alpha n} + \left(1 - \frac{2}{\pi}\tan^{-1}\sqrt{\alpha n}\right) n$$

As the coverage of the test sequence approaches 100%, the square root term will dominate, and again r_{av} is $O(\sqrt{n})$ while fault identification is $O(n)$
Case 3: ($b = 3$, function is not particularly random testable)

$$r_{av} \approx \frac{1}{\alpha\pi}(\alpha n)^{\frac{2}{3}} + \left(1 - \frac{2}{\pi}\tan^{-1}\left((\alpha n)^{\frac{1}{3}}\right)\right) n$$

As the coverage of the test sequence approaches 100%, the first term will dominate, so that r_{av} is $O(n^{\frac{2}{3}})$ while fault identification is $O(n)$. Even in this case, where random vectors are very slow to detect faults, the ratio of r_{av} to n still tends to 0 as n tends to infinity. Naturally, practical tests cannot have infinite length, but nonetheless, the amount of effort required to calculate first failing measurements is much less than that needed for signatures which require full simulation over all vectors.

In conclusion, the use of failing pattern signatures has two major advantages over full simulation. First, in demand-driven applications, large speed-ups can be obtained on average. Second, in dictionary applications, the total amount of effort will be greatly reduced. In both cases, the cost savings will increase with test length.

5.4.2 Failing Output Measurements

The complexity of calculating the failing output set for a given fault and test sequence is equivalent to a fault identification step. Thus, this measure is not practically useful as a predictor. However, a similar measure may be calculated much more simply. This measure is known as the potential failing output set and is defined as follows:

Definition 5.4.1: The *potential failing output set*, $Y_i = \langle y_1, y_2, \ldots, y_m \rangle$, is defined for fault f_i in F_Λ such that y_j is 1 iff there is a path from f_i to output j of Λ

Note that the potential failing output set does not require that the path be active, or even activatable. All that is necessary is that there be a physical connection present. Clearly $Z_i \subseteq Y_i$ where Z_i is the failing output set for fault f_i. Now Y_i can be calculated for a given fault $f_i \in F_\Lambda$ in one pass through Λ of complexity $O(L)$ where L is the number of lines in the circuit. This complexity is equivalent to one simulation pass through the circuit.

In addition, theorem 5.2.4 can be applied to state the following:

Corollary 5.2.4h: The potential failing output set is a predictor for a circuit of 2 or more outputs where there exists faults f_i and f_j such that $Y_i \neq Y_j$.

Thus the potential failing output set is a practical predictor for test lengths greater than the number of lines in the circuit. Such test lengths are typical with random or pseudo-random test vectors

5.4.3 Fixed Length Signatures

By corollary 5.1.2e, the complexity of calculating a signature which depends on all output bits is $O(mn)$. Any practical signature must potentially depend on all output bits, since it is assumed that a fault may potentially be detected by any vector. All signatures which are passive in nature -- those which always require the same number of vectors -- always require mn bit operations to calculate. Examples of this type of compaction method are:

- Signature analysis schemes, including MISR, MICA, single or multiple LFSR, single or multiple CA, and BILBO.
- Counting schemes, including weight and transition counting, spectral coefficients, parity, output data compaction and accumulator compression testing

In fact, the only schemes not included are those such as circular BIST and the failing pattern methods of section 5.4.1. In circular BIST the test set itself depends on the fault. The complexity of this method will depend on the termination condition used, but will be at least $O(lN)$ where l is the length of the shift register and N is the length of the test.

Comparing these complexities to those obtained for failing pattern measurements, it is apparent that conventional signatures cannot match the performance gains of the failing pattern methods, in either demand-driven or dictionary-based diagnosis. Further, the effort involved in obtaining signatures is virtually equivalent to that required for fault identification, making their performance as predictors less useful

5.4.4 Cost Effect of Fault Simulation

Throughout this section results have been obtained assuming that output functions ϕ_i were available for each fault f_i. The complexity of obtaining these has not been addressed, however In section 2 5.6.1 it was observed that the complexity of simulating h distinct faults and n input vectors for an m output circuit is certainly not less than the complexity of fault identification, $O(hmn)$, since each of the mn bits must be generated for each of the h faults This observation resulted in Assumption 2 1 If the simulation effort required to obtain one output vector for one fault is x, then the effort required to produce k output vectors for one fault is kx and the effort required to produce one output vector for each of h faults is hx. Under this assumption, the relative costs in signature calculation remain the same whether fault simulation costs are included or not.

Chapter 6 Structures for Observing Circuit Parameters

Chapter 5 outlined a variety of potential signatures for use with a hierarchical approach to fault location. Obtaining these signatures requires certain physical structures, the construction of which is the subject of this chapter. In light of the requirements outlined in chapter 4, implementation of the structures as part of the chip or module being tested, and hence in the presence of hardware overhead constraints, is a significant concern.

This chapter develops methods of producing the error stream (observed stream exclusive-or'ed with the fault-free stream) required by the error weight count, failing pattern, and failing output signatures. Recall that the complexity of calculating these signatures was shown to be much less than that required by conventional signatures, and that these signatures were shown to be predictors The chapter also demonstrates how error streams can be generated, how the size of both the streams and the generating unit can be reduced when space for testing hardware is at a premium, and describes the hardware used for calculating the signatures themselves.

In hierarchical fault diagnosis, the signatures themselves contain useful diagnosis information. This information is applied sequentially to reduce the set of potential failures from the entire fault set to a more manageable amount using observations known as predictors. The expected count permutation was shown in the previous chapter to be a useful predictor If it is to be used in a hierarchical reduction, detection probabilities are required, and their values must be estimated using some error model The uniform error model, where all such probabilities are assumed to be constant and 0 5, is clearly inappropriate This dissertation shows how both the independent and asymmetric error models (see section 3 1) may be used, and demonstrates good corroboration with experimental results for both models (chapter 9)

Three concerns must be addressed before applications of the hierarchical fault location techniques can be discussed: First, a multiple output circuit contains multiple streams, whereas the signatures of chapter 5 require a single stream. Second, an error stream is not directly available. Finally, physical structures to calculate the signatures have not yet been described. These concerns are addressed in the following sections.

6.1 Parallel to Serial Conversion

A number of options are available for converting the m outputs of a circuit into a single bit stream. These are divided into two categories: Treating a single output vector as m bits within a serial sequence, termed *serial streaming*, and combining the m outputs into a single bit with some initial compaction step, termed *parallel to serial compaction.*

6.1.1 Serial Streaming

In serial streaming, the m outputs of a circuit are loaded into a parallel-load shift register with each test vector, then serially shifted out to be analyzed The advantages of treating a single output vector as part of an output stream are obvious: No information is lost, and the failing output set can be directly observed. There are potential disadvantages to the method, however The amount of data to be investigated in an n vector test is mn bits. In addition, the test cannot be run at circuit speed because of the continual shifting of output bits. The latter may not be a serious problem in scan-design circuits, since the shifting must be performed in any case (although multiple scan chains may cause problems if not treated independently).

Computing the detection probability of a fault in serial streaming is straightforward, since the stream may be considered to comprise m independent streams, each with detection probability equivalent to the value at its associated circuit output This applies to both the independent and asymmetric error models. The detection probability of the entire quotient stream with the independent error model mav be defined as the average detection probability on the m substreams

6.1.1.1 Overhead Considerations

As described above, serial streaming requires that the entire fault-free output be matched with the output of the scan chain for a total of mn bits of information While the benefits of precise signatures are available, their value may still not justify this high storage cost In an effort to avoid this overhead while retaining much of the value of the failing output information, the following hybrid method is proposed

Serial streaming will be performed on the first few vectors, say n_1 (at most a few hundred), after which a selective parity signature on the scan chain will be obtained. This parity signature comprises multiple parity bits, each of which is obtained as follows: The scan chain is shifted into an LFSR, and also through a parity unit, which contains K parity checkers. These are divided into two groups of K_1 and K_2 each, such that K_1 and K_2 are relatively prime. The ith parity checker in group I records errors on bits j such that $i = j \bmod K_I + 1$ for $I \in \{1,2\}$. This scheme will detect all odd errors on a given pattern, all double errors which do not occur at spaces of $K_1 K_2$, and the vast majority of other even errors. If K_1 and K_2 are set to be greater than $\sqrt{m}$, double errors cannot cause cancellation. Splitting K into three or more blocks will further reduce cancellation, but at a higher cost. Two blocks have been chosen as a compromise. The entire scheme is shown in figure 6.1.

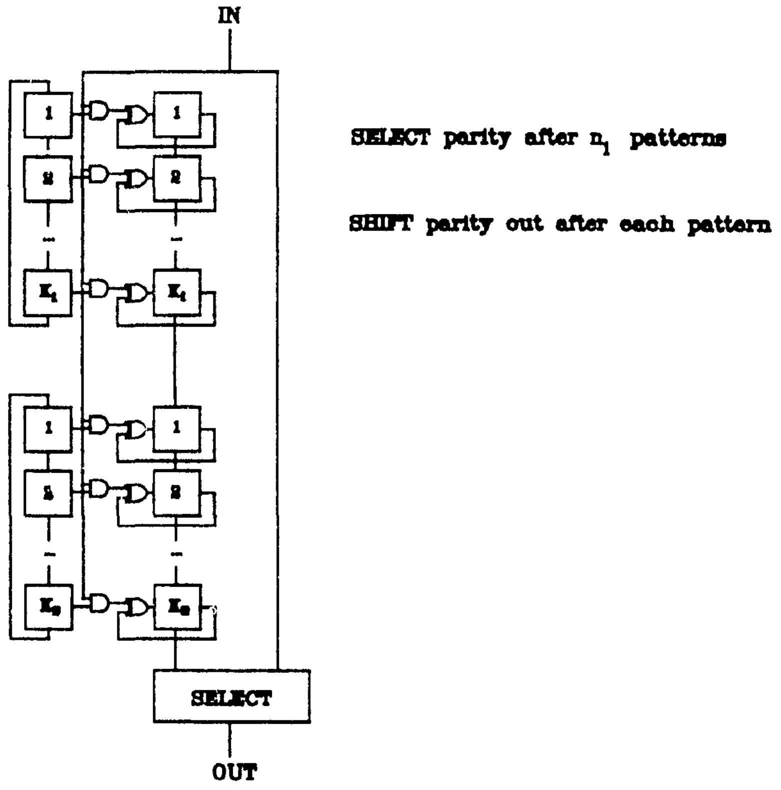

IN
1
2
K_1
1
2
K_2
SELECT
OUT
SELECT parity after n_1 patterns
SHIFT parity out after each pattern

Figure 6.1 Parity Block for Serial Streaming

Figure 6.1 shows a ring counter implementation. Binary counters and demultiplexers could also be used, but are likely to be more complex. The clock signals are not shown, but one is needed per input pattern to increment the counter in the SELECT unit, another is required to shift the scan chain (IN) and the ring counters, and finally a third is required to shift out the parity units after each input pattern (once n_1 patterns have been applied)

When methods other than statistical sampling are used to calculate detection probability in this technique, there may be some resulting inaccuracy because of correlation between circuit outputs. Keeping related outputs on separate parity chains (spaced at multiples of neither K_1 nor K_2 outputs apart) is the easiest solution to this problem Aside from statistical sampling, detection probabilities for the selective parity signature may be calculated using, for example, algorithm 6 1 *

```
{ prob[i] is detection probability of ith parity bit.
  p[j] is detection probability on circuit output j }
for i = 1 to K do
    prob[i] := 0.0.
for j = 1 to m do
    bit1 = j mod K1 + 1;
    prob[bit1] := prob[bit1] + p[j] - 2 * p[j] * prob[bit1];
    bit2 = j mod K2 + K1 + 1;
    prob[bit2] := prob[bit2] + p[j] - 2 * p[j] * prob[bit2];
```

Algorithm 6.1 Example parity detection probability independent error model

6.1.2 Parallel to Serial Compaction

In parallel to serial (P/S) compaction, a group of m outputs is combined into a single bit stream The term compaction is used to reflect potential information loss in this combination. One potential method for performing P/S compaction is a parity tree which has been suggested previously, e g. [Rob87], as a means by which count techniques could be applied to multiple-output circuits Unfortunately, information loss is incurred which could prevent detection of a large number of faults This is caused by correlation between circuit outputs, as exemplified by the circuit of figure 6 2

A fault on line l, or any fault which propagates only through l will never be detected with a parity tree, since the two complementary outputs will cancel each other out Although this is an extreme example, correlation between outputs can have damaging

* Algorithm 6 1 uses the independent error model and is based on COP [Brg84] Its complexity is $O(m)$ The asymmetric model may also be used, as may other testability algorithms which may be better able to account for any correlation in error bits resulting from the parity checkers

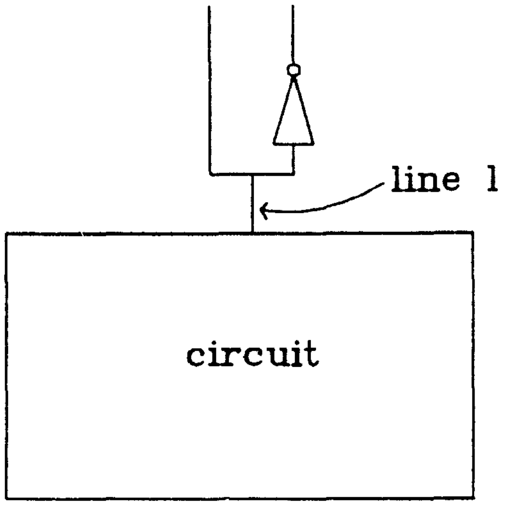


Figure 6.2 Example Fault

effects on detection of other faults, making a parity tree too vulnerable a structure for P/S compaction

Another potential method for performing P/S compaction is a MISR, as used in output data modification (see section 3.5.6). A MISR, however, tends to eliminate any distinctions between detection probability, as shown by the following theorem:

Consider an m-output circuit, to be tested by n input patterns. Each fault has an associated set of detection probabilities $\{p_i \quad 1 \leq i \leq m\}$, where each p_i is the detection probability of the fault on output i *

Theorem 6.1: The random pattern detection probability of a fault on the quotient output of a MISR with feedback tends to $\frac{1}{2}$ with increasing test length provided that there exists at least one $p_i \neq 0$ or 1.

Proof: The random pattern detection probability of a fault on the quotient output of a MISR is the same as the probability of the last stage of the MISR being 1 in the error domain. The results given in [Iva88a] [Iva88b] [Wil89] and [Mil89] show that this latter probability tends to $\frac{1}{2}$ with increasing test length provided that there exists at least one $p_i \neq 0$ or 1

Since detection probability is to be used as a distinguishing attribute, it is necessary to preserve detection probabilities through the P/S compactor. Note that theorem 6.1 requires that the MISR have feedback. If the feedback is removed, the theorem no longer

* Although the independent error model is assumed here a similar result can be obtained for the asymmetric error model, using the methods of [Iva88b] for the generalized error model described in section 3.4.5

applies. In this case, the MINSR (multiple input non-feedback shift register, pronounced "mincer") divides by the degenerate polynomial x^m, and the detection probability of any fault on the quotient stream may be calculated in a straightforward fashion, given detection probability values for each output

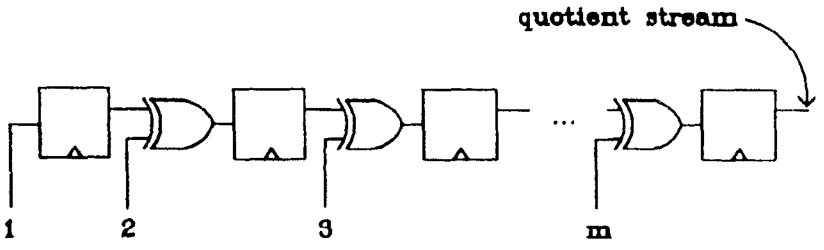


Figure 6.3 MINSR Structure for P S Compaction

The detailed MINSR structure is shown in figure 6.3 By making use of the fact that successive circuit outputs of a combinational circuit are independent when the input vectors are independent, together with some elementary probability theory, the following algorithm for detection probability in the quotient stream under the independent error model is derived:

```
{ prob is detection probability of MINSR quotient stream }
prob = 0.0;
for $i = 1$ to $m$ do
    prob = $p[i]$ + prob − 2 * prob * $p[i]$;
```

Algorithm 6.2 MINSR detection probability, independent error model

Algorithm 6.2 is of complexity $O(m)$ Note that the above probability does not apply to the first $m-1$ bits in the quotient stream, since these are not affected by all outputs (it takes m cycles for output 1 to shift through the register). However, provided that $m \ll n$, this difference should not affect overall results significantly (Another possibility is to apply $m-1$ additional input patterns before examining the MINSR stream)

A similar algorithm may be derived for the asymmetric error model The asymmetric model requires that the fault-free output sequence be known, and hence that the input vectors be fixed before the calculation of the detection probabilities from the MINSR can proceed (the initial calculation of p_D and $p_{\overline{D}}$ for each output does not require this information). Since the fault-free sequence will eventually be required in any case to compute a fault-free signature, the asymmetric model does not require additional simulation, merely that these simulation results be available. Note that these results can come from a functional simulation, before the actual circuit has been laid out. Algorithm 6.3 shows the calculation of MINSR detection probabilities under the asymmetric error model.

```
{ prob[j] is detection probability of jth bit of MINSR stream.}
for $j := 1$ to $n$ do
    prob[$j$] := 0.0;
    if $j < m$ then
        first_out := $m + 1 - j$;     { First output which can affect prob[$j$].}
    else
        first_out := 1;
    for $i :=$ first_out to $m$ do
        if fault_free_output[$i$][$j$] = 1 then
            prob[$j$] := $pD[i]$ + prob[$j$] - 2 * prob[$j$] * $pD[i]$;
        else
            prob[$j$] := $pDbar[i]$ + prob[$j$] - 2 * prob[$j$] * $pDbar[i]$;
```

Algorithm 6.3 MINSR detection probability, asymmetric error model

Note that algorithm 6.3 takes the time lag in probability effects present in the first $m - 1$ vectors into consideration. This additional information is not without cost, however, as the complexity of the algorithm increases to $O(mn)$, as compared to the $O(m)$ of algorithm 6.2, because the probability is calculated separately for each test vector. The combinations of p_D and $p_{\overline{D}}$ on the m outputs can potentially yield 2^m different probability values on the quotient stream. Since 2^m could be greater than the number of test vectors, n, it is possible for every bit on the quotient stream of the MINSR to have a different error probability when the asymmetric error model is used at each circuit output. This provides the physical justification for the generalized error model, which was described in section 3.4.5.

While the amount of data resulting from P/S compaction is considerably less than in serial streaming, the failing output set can no longer be observed on the output stream. The effect of this loss is one of the tradeoffs to be considered when implementing hierarchical fault location.

6.2 Creating the Error Stream

Once detection probabilities for a given fault are available at all outputs, it is a straightforward step to determine its detection probability on the serial stream, using the methods of the previous section At this point, the expected count permutation, for example, of chapter 5 may be used to distinguish between the faults This permutation requires an error stream, which in turn requires fault-free values.

Thus, the fault-free serial stream is required to estimate the detection pro[illegible]ity of an observed fault Note that the fault-free serial stream values are already available from the simulation required to calculate the fault-free signature If serial streaming is used, the amount of information required is up to mn bits for an m-output circuit tested by n vectors (this amount is reduced to $mn_1 + K(n - n_1)$ bits when the parity block is used) Complete storage of the fault-free output eliminates the information savings which help justify data compaction, although this storage is static and does not need to be written during testing If, on the other hand, a MINSR is used, only n bits need be stored, equivalent to the amount of data required for an additional input to the circuit Hence, parallel to serial compaction using a MINSR may be the more suitable technique for obtaining a manageable-sized error stream

The amount of memory required to store a complete output stream may be too large in instances where there are constraints on the amount of hardware overhead. An example of where such constraints might arise is the circuitry's being included on the chip or module being tested

As with the Output Data Modification (ODM) BIST technique, (see section 3 5 6), it is possible to generate error streams on-chip using a PLA, for example, to generate the fault-free stream This fault-free sequence generator or *modifier block* in ODM terminology, is assumed to be the largest overhead in the generation of the error stream,* and its size is related to the number of test vectors As pointed out in [Zor86] and [Zor87], the circuitry required to produce this sequence, assuming a MINSR is used, is a single-output combinational block that need only produce the n bits required to match the MINSR's quotient stream Q, as opposed to the $m2^l$ bits required to specify an l input, m output function. Since $n \ll m2^l$, the modifier block's size can be small in comparison to that of the circuit itself

Since the modifier block is the largest contributor to overhead, reducing its size is the most obvious way to address hardware overhead con[illegible] [illegible] size reduction requires some relaxation of the block's specifications [illegible] in particular removing the requirement for perfect matching of the fault-free quotient sequence The following

* In some cases the P/S conversion circuitry may be larger but it is the sequence generator where size reductions may be obtained

sections examine the partial matching technique and provide some techniques for the construction of modifier blocks. Additional methods may be found in [Zor87].

6.2.1 Partial Matching

ABC	f	g
000	0	0
001	1	1
010	0	0
011	1	1
100	1	1
101	0	1
110	1	1
111	1	1

Table 6.1 Example functions for partial matching

The size of the modifier block may be reduced if it generates a sequence which is not precisely identical to the fault-free quotient stream. For example, consider function f defined in table 6.1. Its minimized sum of products form is:

$$\overline{A}C + A\overline{C} + BC$$

If this was the fault-free output function, then, given only A, B, and C, f would require a 3-input OR gate, 3 2-input AND gates and 2 inverters to implement. On the other hand, changing a single bit of f results in g, whose minimized form is

$$A + C$$

which requires only a 2-input OR gate to implement. Thus, substantial savings in function implementation size are possible with minor changes to the function.

Permitting incomplete matching of the fault-free sequence will result in an error count which is not exactly 0 in the fault-free case since the stream to be analyzed will be only an approximation of the error stream. This will have the effect of narrowing the range of the expected count permutation. The loss in information may result in reduced performance, so matching should always be kept as high as possible, given hardware constraints. In particular, the utility of the failing pattern signatures is severely limited when the observed errors are due to imperfect matching, rather than a circuit failure.

For this reason, it may be desirable to eliminate these signatures in circuits where hardware overhead must be minimized and pay the price in additional effort for fault location. A compromise is to fix the first F bits at a 100% matching, then allow partial matching on the rest.

6.2.2 Constructing a Modifier Block for Partial Matching

In this section, the construction of the modifier block is considered. The modifier block is a single output circuit which is intended to generate an approximation to the error stream for a particular sequence of test patterns. As was pointed out earlier, it can be much smaller than the circuit under test, since its functionality is limited. Numerous methods are possible for the block's construction. These are divided into two major classes: sequential circuits such as autonomous LFSRs, cellular automata, or other finite state machines, and combinational circuits such as PLAs.

6.2.2.1 Sequential Implementations

Although some progress has been made recently in the design of sequential circuits for general applications ([Dev88]), some sort of state diagram is typically needed to implement them. When designing a modifier block, the desired (but not necessarily required) output sequence is available, but both the number and labelling of the internal states are missing. Hence the methods of [Dev88] are not of immediate help. Agarwal [Aga83] and Zorian [Zor87] have investigated some methods of implementing the modifier block as a sequential circuit, and reported a certain amount of success, but neither could achieve a guaranteed performance level for a general function.

In some sense, this lack of results is understandable. Consider a k-stage LFSR, for example. It has 2^k possible initial states and 2^k different feedback connections, resulting in at most 2^{2k} different infinite-length sequences that it can generate. There are 2^n different sequences of length n, so on average an LFSR of length $0.5n$ could be required to generate them. (Note that the preceding should in no way be considered a formal proof). In fact, exact matching of a sequence of n bits requires in the worst case an LFSR of length $n-1$, to generate the sequence 1000 ... 0001. On the other hand, generating this sequence to an accuracy of $\frac{n-2}{n}$ requires only a ground wire.

So, generation of matching sequences with a predetermined level of accuracy by a sequential machine of limited complexity remains an open problem.

6.2.2.2 Combinational Implementations

This section considers the generation of matching sequences by combinational circuits, in particular 2-level combinational circuits, whose inputs are generated by simple sequential machines such as counters, LFSRs, CAs etc. Analysis will be further restricted to autonomous modifier blocks; that is, those which share no inputs with the circuit under test. This aids in the diagnosis of primary input faults, since the effects of these cannot propagate through both the CUT and modifier block. It also allows the modifier block to be physically positioned independent of the circuit under test, and results in the size of the modifier block being determined for the most part by the size of the test, independent of the nature of the circuit under test.

With the above assumptions, the problem of finding a 100% match is reduced to finding the smallest implementation of a given "typical" function. Finding a closed form solution for the average number of terms in a minimized 2-level implementation of such a function remains an open problem [Fle89], although some progress has been made. The initial results in the area were given by Mileto and Putzolu [Mil64], where the average number of k-cubes, prime k-cubes, and essential k-cubes was given.* A 2-level minimization takes at least the essential cubes and then adds additional, preferably prime, cubes until a complete match is obtained. The number of essential cubes provides a very loose lower bound on the average size of such a minimized function, and is of little use when discussing functions with weights close to half their length [Fle89]. The expressions of [Mil64] were used to compute values for average total numbers of cubes, prime cubes, and essential cubes. These are given in table 6.2.

Experiments were performed to determine the complexity of matchings for a variety of different test lengths. The quotient sequences used were from the 74LS181 ALU, which was chosen because the sequences could be generated quickly. Similar results have been obtained with other benchmark circuits given in [Brg85] and described in chapter 9. The experiments proceeded as follows.

First, an exact matching was obtained using ESPRESSO [Bra82] (the second column of table 6.3 gives the number of terms in this minimization) and additionally by McBoole [Dag85], a locally available minimization package (see the third column of table 6.3). Note that the McBoole minimization is slightly better than that of ESPRESSO for

* If an n-variable function is considered to consist of a set of points in an n-dimensional space, a k-cube is a k-dimensional subspace where all points belong to the function. A prime k-cube is not completely contained in any $k+1$-cube while an essential k-cube is a prime k-cube which must be part of any minimized sum of products representation of the function.

Sequence Length	Total Cubes	Prime Cubes	Essential Cubes
8	7	3	2
16	18	5	3
32	41	11	5
64	96	24	7
128	220	53	10
256	503	118	13
512	1141	262	16
1024	2572	584	20
2048	5766	1304	24
4096	12858	2902	30
8192	28542	6436	36

Table 6.2 Average Numbers of Cubes for Various Test Lengths

those functions examined. This supports claims made in [Dag85] regarding functions with fewer than 20 inputs. Because of ESPRESSO's wider availability, however, its minimization was selected for approximation.

Next, the approximate matching algorithm (algorithm 6.4) was used to give matchings of 90% and 75% The number of minterms in these matchings are given under the partial matching heading in table 6.3.

Comparing tables 6.2 and 6.3 shows that, if the quotient sequences are taken to be "average" functions, none of the quantities in table 6.2 gives a good estimate of the average number of cubes in the ESPRESSO minimization.

Algorithm 6.4 is a fairly straightforward greedy approach. Its initial input data is a minimized sum of products representation of a function It picks a cube at random from the current representation, then finds the cube which most closely matches the selected cube, in terms of overlap These two cubes are then merged into a single cube Matching terms in the two cubes are left alone, while mismatching ones are set to "don't care" with 50% probability, or 0 or 1 with 25% probability each Several random matchings are tried with the given cube, and the one which causes the least change to the function is retained. The process is tried for several initial cube selections The combination of cubes which produces the least disruption to the original function is chosen and the new merged cube is inserted into the function description in place of the other two This process continues until the coverage of the original function drops below a preselected

Stream Length	Number of Cubes			
	100% matching		Partial matching	
	ESPRESSO	McBoole	90%	75%
8	2	2	2	1
16	3	3	2	2
32	9	9	7	3
64	15	15	11	4
128	24	23	20	6
256	49	48	35	17
512	95	93	66	30
1024	174	165	115	50
2048	326	315	230	99
4096	610	593	421	185
8192	1175	1165	800	410

Table 6.3 Size of Modifier Circuit for Various Test Lengths

level

The complexity of the algorithm is $O(c^2 \log n)$ where c is the number of cubes in the minimized representation and n is the test length (Each cube is of length $\log n$). Because there is no backtracking in the algorithm, a globally optimal solution is unlikely to be achieved. Thus, algorithm 6.4 is certainly not the best possible for developing approximate matchings. Using better heuristics, or allowing the possibility of backtracking, could probably produce better matchings with fewer cubes

While algorithm 6.4 is able to reduce the number of cubes by about one third for 90% matching and by two thirds for 75% matching, the size of the modifier circuits may still be somewhat large. However, recent work by Vasudevamurthy [VaJ89] has shown that multilevel implementations of two-level minimizations can significantly reduce circuit complexities, often by a factor of two or three. Future investigations could focus on the combination of multilevel and approximate matching strategies in an effort to synthesize much smaller modifier blocks than those developed so far

6.3 Signature Hardware

This section describes the hardware needed for obtaining various signatures outlined in chapter 5, given an error stream. The signatures considered are the error weight count, failing pattern signatures, and failing output signatures.

```
{Merge cubes until coverage falls below specified value.}
coverage = 100%;
while coverage > MINCOVERAGE
begin
    tries = 0,
    repeat
        if (tries mod TRYEACHCUBE) = 0
        begin
            select cube C at random;
            for each remaining cube X
            begin
                let y = compare(C, X);
                let M = X for the maximum y;
            end
        end
        let R = merge(M, C),
        let coverage = matching with R instead of C and M,
        let R1 = R, M1 = M for maximum coverage,
        tries = tries + 1;
    until tries > MAXTRIES;
    replace M1 and C with R1;
end

integer function compare(cube1, cube2)
    compare = 0;
    for each variable i in cube1, cube2
        if cube1[i] = '-' or cube2[i] = '-'
            compare = compare + 1;
        else if cube1[i] = cube2[i]
            compare = compare + 2;
    return compare;

cube function merge(cube1, cube2)
    for each variable i in cube1, cube2
        if cube1[i] = cube2[i]
            merge[i] = cube1[i];
        else if random(2) = 1
            merge[i] = '-'.
        else if random(2) = 1
            merge[i] = cube1[i].
        else
            merge[i] = cube2[i];
    return merge.
```

Algorithm 6.4: 2-level approximate matching

6.3.1 Error Weight Count

Once an error stream is available, an error weight count may be obtained by simply counting the number of 1s on the error stream. Thus, any form of counter will suffice, provided that it is able to count up to the length of the error stream. If the counter is included on chip, it should be part of a signature scan chain in order to be able to report its final contents to the outside world.

6.3.2 Failing Pattern Signatures

Obtaining failing pattern signatures is similar to counting the weight of the error stream. A first failing pattern may be identified by incrementing a counter until a 1 occurs on the error stream, then terminating.

A kth failing pattern signature is somewhat more complex, in that two counters are required -- one to count clock cycles (one for each bit on the error stream) and a second to count k errors and then signal the first to stop The second counter need only be $\log k$ bits long, and hence presumably smaller than the first ($\log n$ bits with a MINSR providing the error stream), and only the position of the kth error need be reported to the outside world It may be desirable with a kth fail counter to report the total number of errors observed over the test length in the event that this number is less than k. In this case, the contents of the k counter would form the signature. No flag would be required to distinguish between the two, since if the signature was the number of errors its value will be less than k, while the position of the kth failing pattern must always be at least k.

A last failing pattern signature requires a counter and a register. The counter counts clock cycles, and the register is loaded from the counter whenever a 1 appears in the error stream. The register value forms the signature.

Note that if several such signatures were included in one unit, they could share the same clock cycle counter and load their signature registers from this counter when required

6.3.3 Failing Output Set

A failing output register cannot be employed when a MINSR is used for P/S compaction, since output information is no longer present, and its utility is limited in the presence of partial matching However, when used in conjunction with serial streaming, the failing output signature is obtained as follows:

An m bit counter is used to determine which of the m outputs is active. A binary counter with a demultiplexer may be used, but an m bit ring counter may be a simpler implementation. Whenever a 1 appears on the error stream, the corresponding failing output register bit is set. When a parity block is used to reduce the size of the error stream, the first n_1 vectors should use an m bit register, while the remaining ones should be diverted to a K bit recorder. The signal for this conversion may be shared with the parity block.

Chapter 7 Applications of Hierarchical Fault Diaguosis

This chapter describes sample methods of applying hierarchical fault location techniques to circuits tested with random or pseudo-random vectors and whose output values are compacted into several signatures. The techniques exploit information present in the circuit's signature in order to reduce the potential fault set at each of several steps using some of the predicting techniques described in chapter 5 and the hardware described in chapter 6. In addition, they possess the characteristics outlined in chapter 4 which are required of such fault location methods signatures which contain location information, small fault dictionary size, single-fault resolution, ability to produce a dictionary in a demand-driven fashion, applicability to multi-output combinational circuits, and potential hardware implementation

Three sample methods are described in this chapter. These are known collectively as DAPPER and individually as Standard DAPPER, Scan DAPPER and k-Max DAPPER. All employ first-failing pattern information and the expected count method of determining detection probability, which was described in previous chapters Scan DAPPER also employs the potential failing output set, while k-Max DAPPER modifies the expected count method by using the kth failing pattern These signatures were shown in chapter 5 to be both predictors and easily calculated for modelled faults, making them the most suitable of the signatures investigated.

This chapter extends the theoretical foundations of the DAPPER method and provides the practical details of their implementation when applied to random compact testing (RCT) and built-in self-test (BIST) Distinctions between the two facets of the RCT environment are made when necessary, usually with regard to overhead considerations and the resulting partial matching for built-in self-diagnosis circuitry The chapter concludes with an examination of the aliasing characteristics of the methods and how they relate to multiple fault models.

Fault location may be performed using a combination of signature information and post-test simulation, or by constructing a dictionary It is further assumed that once the test is complete, no record of the actual output sequence remains apart from the signatures collected during data compaction The three DAPPER methods are now investigated in detail.

7.1 Standard DAPPER

Standard DAPPER uses two hierarchical reductions in fault diagnosis and requires three signatures An error weight count signature gives an estimate of fault detection probability, which provides coarse resolution and quickly eliminates many potential faults from consideration. In addition, a first-fail counter is included to further reduce the fault set and increase resolution These two signatures are predictors (see chapter 5), and are thus able to reduce the effort required for fault location without a dictionary. Finally, to ensure fine resolution, an LFSR signature is comput d. The complete scheme is shown in figure 7.1.

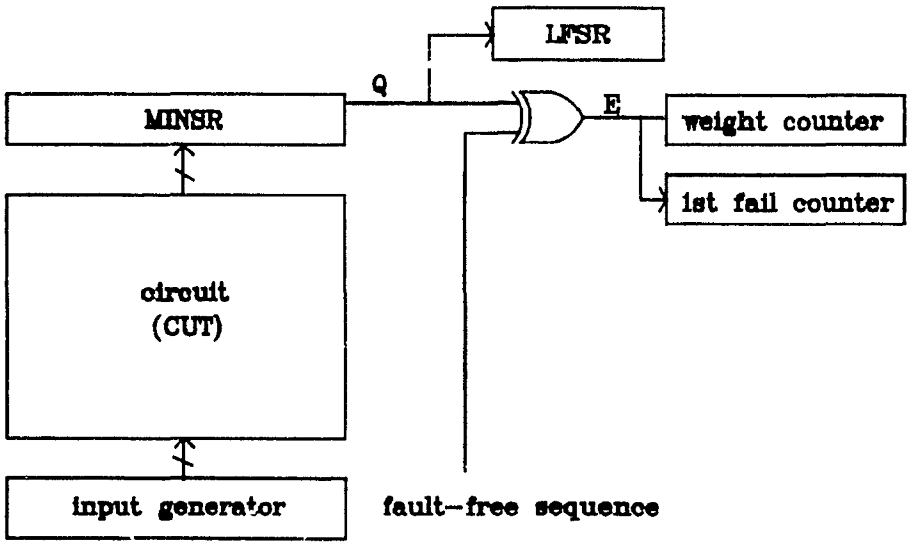


Figure 7.1 The DAPPER Method

The Standard DAPPER diagnosis procedure when no dictionary is to be constructed is as follows. Detection probability on the quotient stream Q is calculated for each fault (modifying this stream by taking its exclusive-or with the fault-free stream does not affect detection probability since an exclusive-or gate is guaranteed to propagate an error which appears at only one of its inputs), using algorithm 6.2 for the independent error model, or algorithm 6.3 for the asymmetric error model. These probabilities are stored as a diagnosability profile, which is a list, sorted by detection probability, of each fault and its detection probability.

For a given circuit under test (CUT), the quotient stream output, Q, (from the MINSR) is fed to an LFSR, and also exclusive-or'ed with the fault-free stream, yielding an error stream, E.* The weight (number of 1s) of this stream is tallied in the weight counter, while the second counter records the position of the first failing pattern (first-fail). The weight value provides an estimate of the detection probability of any fault present. The accuracy of this estimate will increase with test length.†

Since several faults may have similar detection probabilities, post-test simulation is used to distinguish between them. Simulation is performed starting with the fault whose predicted detection probability is closest to that observed and continues until the fault is found. With the first-fail information available, simulation can stop when either the simulated fault fails on a pattern before the observed first failing pattern, or the simulated fault does not fail on the observed first-fail. If the first-fails match, simulation is performed on all vectors and the signatures are compared. If the weight, first-fail and LFSR signatures match, the fault has been located.

There are a few circumstances where this method may not result in unique (to fault equivalence) fault isolation. These forms of aliasing, where two unequivalent faults have identical signatures, are discussed in section 7.4.

7.1.1 Resolution With Detection Probability

As noted in chapter 5, the observed weight is only an estimator of the detection probability, rather than an exact reflection of it. So, given a detection probability and an observed weight, a *p-value* (see [Won77] or any other statistics text for more information on p-values) can be obtained. A p-value in this case gives the probability that a fault with a given detection probability would produce an observed weight at least as far from its expected value as the one observed. The concept is explored in detail below, for both the independent and asymmetric error models.

7.1.1.1 Independent Error Model

The expected weight, $E(w_f)$, of a fault f with detection probability p_f on a modifier

* The LFSR signature may also be obtained on stream E when DAPPER is not implemented in hardware. Its position on Q aids in diagnosis of failures in the test hardware itself, as shown later.

† The expected first-fail value is also related to detection probability, but its accuracy is poor and virtually unrelated to test length.

stream of length n with fault-free weight 0 is given by:

$$E(w_f) = np_f \tag{7.1.1}$$

The errors are assumed to be binomially distributed, so the variance of w_f, $\mathrm{VAR}(w_f)$ is:

$$\mathrm{VAR}(w_f) = np_f(1 - p_f) \tag{7.1.2}$$

Because the variance is directly proportional to n, the standard deviation of an observed count with respect to the expected count will decrease as $\sqrt{n}$:

$$\text{S-DEV}(w_f) = \sqrt{np_f(1 - p_f)} \tag{7.1.3}$$

The binomial distribution of errors also allows the standard deviation and expected value of w_f to be used in the application of the Central Limit Theorem of statistics (see for example [Won77], p. 151) to state the following:

Theorem 7.1: The probability of a given fault f, with expected weight $E(w_f) = \mu$ and standard deviation $\text{S-DEV}(w_f) = \sigma$, $\sigma \neq 0$, producing an observed weight at least as far from $E(w_f)$ as v is asymptotically:

$$P(|w_f - E(w_f)| \geq v) = R\left(\frac{|v - \mu| - 0.5}{\sigma}\right)$$

where $R(z)$ is the probability of a standard normal random variable being at least z standard deviations from the mean (e.g. $R(2) \approx 0.046$).*

Proof: By Central Limit Theorem (includes continuity correction for binomial).

The theorem does not apply when $\sigma = 0$, which occurs only when p_f is 0 or 1. In these cases, the sequence is deterministic and the final weight is guaranteed to be np_f.

The values obtained by applying theorem 7.1 are often called "p-values" [Won77]. As an example of these, consider a test of length 2048, with fault-free counter value 0, and a modelled fault a with detection probability of 0 1:

by (7.1 1)	$E(w_a) = 205$
by (7.1.3)	$\text{S-DEV}(w_a) = 13.6$

* $R(z)$ is formally defined as.

$$R(z) = \frac{2}{\sqrt{2\pi}} \int_{|z|}^{\infty} e^{\frac{-t^2}{2}} dt$$

Theorem 7.1 may now be used to determine the probability that observed weight W resulted from fault a:

$$\begin{aligned}
W = 200, \quad & P(|w_a - 205| \geq 5) &= R(0.368) &= 0.718 \\
W = 175, \quad & P(|w_a - 205| \geq 25) &= R(2.210) &= 0.0272 \\
W = 275, \quad & P(|w_a - 205| \geq 70) &= R(5.156) &= 2.53 \times 10^{-7}
\end{aligned}$$

Hence, given any observed faulty weight, the p-values for each fault may be calculated. Each p-value provides an estimate of the likelihood that a particular fault could have caused the observed output I the previous example, it is very likely that fault a would produce a weight of 200, it is somewhat likely that it would produce a weight of 175, and it is highly unlikely that it would produce a weight of 275 Sorting by p-value gives the expected count permutation of chapter 5

The expected behaviour of the first fail counter may be determined in a similar manner Its expected value, r_f, for fault f is also related to the fault's detection probability p_f, as follows· (As a first approximation, consider an infinite test length, so that r_f will have some value for $p \neq 0$.)

$$\begin{aligned}
E(r_f) &= \sum_{i=1}^{\infty} p_f \cdot (1 - p_f)^{i-1} \cdot i \\
&= \frac{1}{p_f}
\end{aligned} \tag{7 1 4}$$

The variance of r_f is given by:

$$\begin{aligned}
\text{VAR}(r_f) &= \sum_{i=1}^{\infty} p_f \cdot (1 - p_f)^{i-1} \cdot \left(i - \frac{1}{p_f}\right)^2 \\
&= \frac{1 - p_f}{p_f^2}
\end{aligned} \tag{7.1.5}$$

where $p_f \neq 0$. When a finite test length is introduced the summation is performed only to n rather than infinity, which complicates matters in that some value for r_f must be assigned when f is never detected among the n vectors In a physical implementation r_f will be n, so that value will be used here, giving

$$\begin{aligned}
E(r_f) &= \sum_{i=1}^{n} \left[p_f \cdot (1 - p_f)^{i-1} \cdot i\right] + (1 - p_f)^n \cdot n \\
&= \frac{1}{p_f} - \frac{(1 - p_f)^n}{p_f}
\end{aligned} \tag{7 1 6}$$

The effect of test length on the expected value of r_f can thus be seen to be both minimal and decreasing as n increases; i.e., (7.1.4) is a good upper bound for (7.1.6) whose accuracy increases with n. Using the above expected valu and solving for the variance gives the following result:

$$\begin{aligned}\mathrm{VAR}(r_f) &= \sum_{i=1}^{n}\left[p_f\cdot(1-p_f)^{i-1}\cdot\left(i-\frac{1}{p_f}+\frac{(1-p_f)^n}{p_f}\right)^2\right] \\ &\quad + (1-p_f)^n\cdot\left(n-\frac{1}{p_f}+\frac{(1-p_f)^n}{p_f}\right)^2 \\ &= \frac{1-p_f}{p_f^2}-\frac{(2n-1)(1-p_f)^n}{p_f}-\frac{(1-p_f)^{2n}}{p_f^2}\end{aligned} \tag{7.1.7}$$

where $p_f \neq 0$, and again the effect of the error term is minimal and decreasing, so that (7.1.5) provides a good upper bound for (7.1.7).

7.1.1.2 Asymmetric Error Model

When the asymmetric error model is used at the outputs, the quotient stream of the MINSR follows the generalized error model. Recall from section 3.4.5 that each output bit has a unique error probability, p_{f_i}, $1 \leq i \leq n$, in the presence of fault f. Since the expected value and variance of the sum of independent binomial streams can be obtained by summation, the expected weight is given by:

$$E(w_f) = \sum_{i=1}^{n} p_{f_i} \tag{7.1.8}$$

and its variance is:

$$\mathrm{VAR}(w_f) = \sum_{i=1}^{n} p_{f_i}(1-p_{f_i}) \tag{7.1.9}$$

Note that (7.1.9) and (7.1.8) reduce to (7.1.2) and (7.1.1) respectively if all the p_{f_i}s are equal. The asymptotic result of theorem 7.1 can still be used for this model, since the Central Limit Theorem continues to apply, provided that $p_{f_i} \neq 0, 1$ infinitely often; that is, there is some randomness in the sequence. So given detection probability values for each fault, p-values can be obtained for any observed weight. When post-test simulation is performed, faults with the largest p-values will be simulated first. Note that the expected value and variance expressions now require a summation and thus obtaining p-values is more complex than with the independent error model by a factor of about $O(n)$.

The equations for the behaviour of the first failing pattern are given by:

$$E(r_f) = \sum_{i=1}^{n} \left[i \cdot p_{f_i} \cdot \prod_{j=0}^{i-1} (1 - p_{f_j}) \right] + n \prod_{j=0}^{n} (1 - p_{f_j}) \qquad (7.1.10)$$

and

$$\mathrm{VAR}(r_f) = \sum_{i=1}^{n} \left[(i - E(r_f))^2 \cdot p_{f_i} \cdot \prod_{j=0}^{i-1} (1 - p_{f_j}) \right] + (n - E(r_f))^2 \cdot \prod_{j=0}^{n} (1 - p_{f_j}) \quad (7.1.11)$$

where p_{f_0} is defined to be 1. Again, little can be said about the solutions to the above equations without some idea as to what the probability values are.

7.1.2 Modifications with Partial Matching

When less than 100% matching of the fault-free sequence is used to create an approximate error stream, the fault-free weight in the counter, w, will be nonzero. A stream with a matching rate of M, $0 \leq M \leq 1$, will result in a value of w equal to

$$w = (1 - M)n \qquad (7.1.12)$$

The results of section 7.1.1 are updated here to take imperfect matching into consideration.

7.1.2.1 Independent Error Model

The possibility of a nonzero weight count in the fault-free case results in a change to the expected weight $E(w_f)$ of fault f with detection probability p_f. Since errors in the independent error model follow the binomial distribution, np_f errors of probability p_f are expected in n bits. Of these, $w p_f$ are expected to be $1 \rightarrow 0$ errors, while $(n - w)p_f$ are expected be $0 \rightarrow 1$. Thus, the expected weight is:

$$\begin{aligned} E(w_f) &= w - wp_f + (n - w)p_f \\ &= w + (n - 2w)p_f \end{aligned} \qquad (7.1.13)$$

The variance of w_f, $\mathrm{VAR}(w_f)$ remains unchanged

$$\mathrm{VAR}(w_f) = np_f(1 - p_f) \qquad (7.1.14)$$

Again, because the variance is directly proportional to n, the standard deviation of an observed count with respect to the expected count will decrease as $\sqrt{n}$

$$\text{S-DEV}(w_f) = \sqrt{np_f(1 - p_f)} \qquad (7.1.15)$$

Since the distribution remains binomial, theorem 7.1 continues to apply

As an example of the changes which can occur with imperfect matching, consider a test of length 2048, with fault-free weight 95, and a modelled fault a with detection probability of 0.1.

by (7.1.13)	$E(w_a) = 604$
by (7.1.15)	$\text{S-DEV}(w_a) = 20.2$

Theorem 7.1 may now be used to determine the probability that observed weight W resulted from fault a:

$$W = 600, \quad P(|w_a - 604| \geq 4) = R(0.173) = 0.868$$

$$W = 650, \quad P(|w_a - 604| \geq 46) = R(2.254) = 0.0243$$

$$W = 500, \quad P(|w_a - 604| \geq 104) = R(5.128) = 2.94 \cdot 10^{-7}$$

Hence, given any observed faulty weight, the p-values for each fault can still be calculated. Each p-value provides an estimate of the likelihood that a particular fault could have caused the observed output. In the previous example, it is very likely that fault f would produce a weight of 600, it is somewhat likely that it would produce a weight of 650, and it is highly unlikely that it would produce a weight of 500

When less than 100% matching of the fault-free sequence is permitted, the expected first-fail value becomes (for $p_f \neq 0$):

$$\begin{aligned} E(r_f) &= \sum_{i=1}^{F} \left[p_f \cdot (1-p_f)^{i-1} \cdot i\right] + (1-p_f)^F \cdot F \\ &= \frac{1}{p_f} - \frac{(1-p_f)^F}{p_f} \end{aligned} \qquad (7.1.16)$$

where F is the first pattern where perfect matching with the fault-free sequence is not provided. The variance is given by.

$$\begin{aligned} \text{VAR}(r_f) &= \sum_{i=1}^{F} \left[p_f \cdot (1-p_f)^{i-1} \cdot (i - E(r_f))^2\right] + (1-p_f)^F \cdot (F - E(r_f))^2 \\ &= \frac{1-p_f}{p_f^2} - \frac{(2F-1)(1-p_f)^F}{p_f} - \frac{(1-p_f)^{2F}}{p_f^2} \end{aligned}$$

(7.1.17)

Clearly, the choice of F could have some effect on both the expected value and variance of the first failing pattern. The larger F is, the less pronounced this effect will be Expressions (7.1.16) and (7.1.17) reduce to (7.1.6) and (7.1.7) respectively in the case of perfect matching with the fault-free sequence, where F is defined to be n.

7.1.2.2 Asymmetric Error Model

As with the independent error model, the possibility of a non-zero fault-free counter weight and a first-fail which is not determined by error alone affects the expectation and variance expressions under the asymmetric error model.

Let $Y = \langle y_i \rangle$, $1 \leq i \leq n$ be the fault-free output sequence from the modifier block (see section 6.2.1) The expected weight is then given by

$$E(w_f) = \sum_{i=1}^{n} a_i \tag{7.1.18}$$

where

$$a_i = \begin{cases} p_{f_i}, & \text{if } y_i = 0, \\ 1 - p_{f_i}, & \text{otherwise.} \end{cases}$$

while its variance remains unchanged.

$$\mathrm{VAR}(w_f) = \sum_{i=1}^{n} p_{f_i}(1 - p_{f_i}) \tag{7.1.19}$$

Note that (7.1.18) and (7.1.19) reduce to (7.1.13) and (7.1.14) respectively if all the p_{f_i}s are equal. The asymptotic result of theorem 7.1 can still be used for this model, since the Central Limit Theorem continues to apply, provided that $p_{f_i} \neq 0, 1$ infinitely often, that is, there is some randomness in the sequence Again, calculation of p-values is more complex than in the case of the independent error model

The modified equations for the behaviour of the first failing pattern are:

$$E(r_f) = \sum_{i=1}^{F} \left[i \cdot p_{f_i} \cdot \prod_{j=0}^{i-1} (1 - p_{f_j}) \right] + F \cdot \prod_{j=0}^{F} (1 - p_{f_j}) \tag{7.1.20}$$

and

$$\mathrm{VAR}(r_f) = \sum_{i=1}^{F} \left[(i - E(r_f))^2 \cdot p_{f_i} \cdot \prod_{j=0}^{i-1} (1 - p_{f_j}) \right] + (F - E(r_f))^2 \cdot \prod_{j=0}^{F} (1 - p_{f_j}) \tag{7.1.21}$$

where p_{f_0} is defined to be 1 and F is defined as

$$F = \min(i) \text{ such that } y_i = 1$$

The effect of the choice of F on the results depends on the individual probabilities although, in general, increasing F will result in a more useful first-failing pattern signature.

7.1.3 Diagnosing Faults Within the DAPPER Hardware

The inclusion of test hardware on the chip or board under test poses a complication not present in random compact testing (RCT) – the problem of faults within the test circuitry itself (In RCT it is assumed that the test hardware is tested independently of the devices being tested and that consequently at test time the RCT hardware is fault-free) In previous work such faults have been ignored, or it has been postulated that these faults will cause catastrophic and hence easily visible failures of the circuit Such claims may not apply to the modifier block in DAPPER, since faults there will merely change the values of the weight count and first-fail It is now shown that DAPPER is able to diagnose these faults, provided that they do not occur in conjunction with other failures in the CUT If faults occur in both the CUT and DAPPER hardware, the effects may not be determinable and fault location may be impossible

Single faults within the modifier block will change the counts, which permits their location Since the LFSR signature is tallied on the unmodified stream, it will remain at its fault-free value in the event of a failure within the modifier block Thus, when a fault-free LFSR signature is observed together with failing weight and first-fail values, two possibilities exist There may be a fault in the CUT which has caused the LFSR to alias, or there may be a fault in the modifier block. In either case, an estimate of the detection probability of the responsible fault is available in the weight and first-fail counters and diagnosis of both possibilities can proceed in the usual manner If the chances of LFSR aliasing are minimized through the use of a sufficiently long register, then a modifier fault may be the most likely candidate

Since only the weight count and first-fail values of a modifier fault are available, there may be an increased chance of signature aliasing When this happens, however, the circuit will appear fault-free, and since the CUT is also fault-free, no serious problem should result from the incorrect diagnosis

While single-fault resolution of modifier faults may be unavailable because of the lack of an LFSR signature, the ability to diagnose faults within itself is a useful addition to the properties of hardware implementations of DAPPER

7.2 Scan DAPPER

Scan DAPPER is identical to Standard DAPPER, except that serial streaming is used for parallel to serial conversion, and the failing output set is obtained. The term Scan DAPPER refers to the fact that the scanning out of output vectors required by

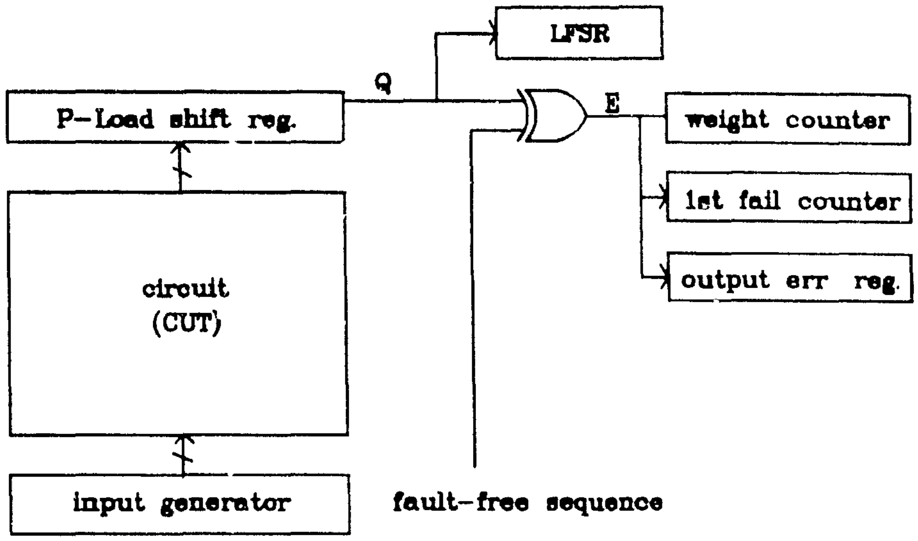


Figure 7.2 Ideal Scan DAPPER, $n_1 = n$

serial streaming will occur anyway in scan design circuits.

The availability of the failing output set will allow a given fault to be eliminated from consideration if an output which is not part of the fault's potential failing output set actually recorded an error. This reduction will allow for easy elimination of many faults. A diagram of the method is given in figure 7.2.

The behaviour of the other signatures may be determined in a straightforward fashion from the descriptions of section 7.1.1. The average detection probability may be used, or for additional accuracy, the m substreams can be treated separately, and their results recombined to give expected counts and first-fails.

7.2.1 Overhead Considerations

As was pointed out in section 6.1.1, serial streaming requires that the entire fault-free output be matched with the output of the scan chain for a total of mn bits of information. While the benefits of accurate first-fail and failing output signatures are available, their value may still not justify this high storage cost. In an effort to avoid this overhead while retaining much of the value of the failing output information, the parity unit described in section 6.1.1.1 will be employed. This set-up is shown in figure 7.3.

When determining expected first-fail values for this hybrid technique, the two regions must be treated separately. When the first failing pattern occurs in the first n_1 vectors, the exact output on which the error was recorded is available. After that, a set

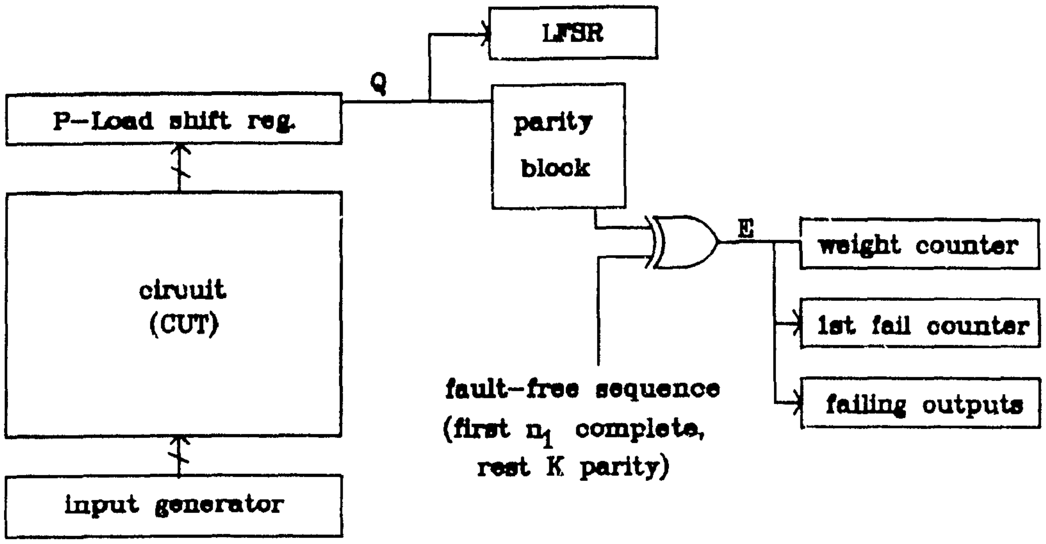


Figure 7.3 Scan DAPPER, $n_1 \ll n$

of possible outputs results from the parity chains.

7.2.2 Modifications with Partial Matching

The overheads required by Scan DAPPER are potentially large: n_1 bits matched completely, followed by a parity compaction of the remaining $n - n_1$ bits, for a total size of $mn_1 + K(n - n_1)$ bits to be matched. Since n is assumed to be much greater than n_1, it is K which is likely to strongly affect this total. (Recall that the K bit parity sequence is formed by two separate parity compactions of length K_1 and $K_2 = K - K_1$ bits each, where K_1 and K_2 are relatively prime.) The overheads involved require that both K and n_1 be minimized in order to reduce the stream down to a manageable size when using DAPPER in self-testing scannable circuits.

This can be accomplished by reducing n_1 to 0, resulting in a signature made up of only the multiple parity bits, as shown in figure 7.4. The scan chain is shifted into an LFSR, and also through a demultiplexer parity unit.

The failing output register now records only blocks of failing outputs (those recorded through each of the K parity checkers), and as a result may not be included if space is truly at a premium.

7.3 k-Max DAPPER

In any system using random input vectors, the presence of random-pattern resistant faults may require long test lengths. In addition, in both Standard and Scan DAPPER,

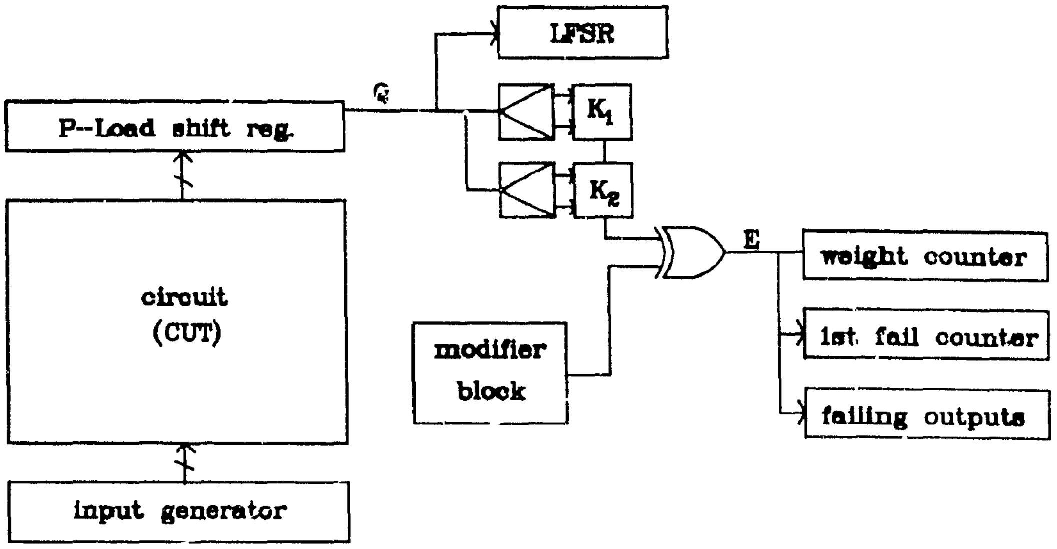


Figure 7.4 Scan DAPPER with Hardware Constraints, $n_1 = 0$

if the first-fail values for an observed and suspected faults match, an LFSR signature must be verified, which requires that simulation be performed over the entire test length Thus, one potential difficulty with DAPPER is that while many faults may not require a long test length for diagnosis, test length is set at a value required by the most difficult-to-detect faults

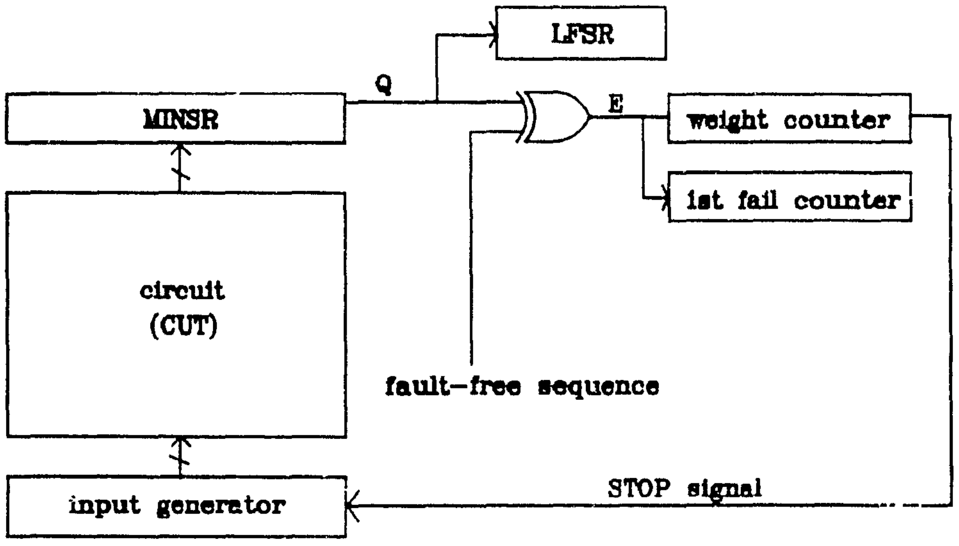


Figure 7.5 k-Max DAPPER

This section proposes a solution, called k-max DAPPER Since a fault can typically be located after it has been detected several times, say k, there is no point in continuing the test after this value has been attained by the weight counter. Thus after the kth failing pattern, the counter reaches k, signals the test pattern generator to stop, and the test is complete The variable in this case is not the weight, but the test length, which is available from the position at which the generator stopped. The test pattern generator will also stop after it reaches n, the maximum length matched by DAPPER. A diagram of the scheme is shown in figure 7.5.

If the entire test is completed, fault location proceeds as in Standard DAPPER. On the other hand, if the counter has reached k, then there is no longer a question of expected weight, but rather expected test length. Thus, the results of section 7.1.1 must be adapted somewhat.

7.3.1 Estimating Test Lengths

The essential premise of the weight counter, that it contains an estimate of detection probability, remains unchanged in k-Max DAPPER. However, the means of estimating it changes For the independent error model, the expected count is no longer test length times detection probability, but rather the expected test length is maximum count divided by detection probability. This concept is now explained more formally.

7.3.1.1 Independent Error Model

In section 7.1.1, it was observed that

$$E(w_f) = np_f \tag{7.1.1}$$

for a fault with detection probability p_f and test length n. When the test pattern generator in k-Max DAPPER has stopped before it reaches the nth pattern, however, w_f is known to be k and it is n_f which is unknown.

Recall from section 7.1.1 that for a test of arbitrary length the expected position of the first failing pattern $r_f^{(1)}$ is given by:

$$E(r_f^{(1)}) = \frac{1}{p_f} \tag{7.1.4}$$

Clearly this equation does not apply when p_f is 0 In this case the fault will never be detected and the counter cannot reach 1, let alone k. Since each output bit is independent, the expected distance of the $i+1$th failing position, $r_f^{(i+1)}$, from the ith failing position, $r_f^{(i)}$, for $i \geq 1$ is also given by

$$E(r_f^{(i+1)} - r_f^{(i)}) = \frac{1}{p_f} \tag{7.3.1}$$

Hence, the expected value of the kth failing position, $E(n_f)$, is given by:

$$\begin{aligned} E(n_f) &= E(r_f^{(1)}) + \sum_{i=2}^{k} E(r_f^{(i+1)} - r_f^{(i)}) \\ &= \frac{1}{p_f} + \sum_{i=2}^{k} \frac{1}{p_f} \qquad (7.3.2) \\ &= \frac{k}{p_f} \end{aligned}$$

Again, since the output bits are independent, the variance of the sum, VAR(n_f), can be obtained from the variance of the first-failing pattern, VAR(r_f):

$$\text{VAR}(r_f) = \frac{1 - p_f}{p_f^2} \qquad (7.1.5)$$

thus,

$$\begin{aligned} \text{VAR}(n_f) &= k \times \text{VAR}(r_f) \\ &= \frac{k(1 - p_f)}{p_f^2} \qquad (7.3.3) \end{aligned}$$

and

$$\text{S} - \text{DEV}(n_f) = \sqrt{\frac{k(1 - p_f)}{p_f^2}} \qquad (7.3.4)$$

The standard deviation and expected value of n_f again allow the application of the Central Limit Theorem in obtaining p-values.

Theorem 7.2: The probability of a given fault f, with expected test length $E(n_f)$ μ and standard deviation S-DEV$(n_f) = \sigma$, $\sigma \neq 0$, producing an observed length at least as far from $E(n_f)$ as v is asymptotically:

$$P(|n_f - E(n_f)| \geq v) = R\left(\frac{|v - \mu| - 0.5}{\sigma}\right)$$

where $R(z)$ is the probability of a standard normal random variable being at least z standard deviations from the mean.

Proof: By Central Limit Theorem (includes continuity correction for binomial)

The theorem does not apply when $\sigma = 0$, which occurs only when p_f is 1. In these cases, the sequence is deterministic and the final length is guaranteed to be k

7.3.1.2 Asymmetric Error Model

As was observed in the previous section, the expected test length for a fault in k-Max DAPPER under the asymmetric error model can be obtained by a summation of k expected first fail values. The equations are somewhat more cumbersome, since closed form solutions are unavailable without individual probability values, but the net result is that expected test length $E(n_f)$ is:

$$E(n_f) = \min j \quad \text{such that} \quad \sum_{i=1}^{j} p_{f_i} \geq k \tag{7.3.5}$$

Calculation of variance is much more difficult. It is more reasonable to rank faults by expected test length and select them in that order than to attempt to apply theorem 7.2 and obtain p-values. Using this technique will no longer necessarily result in a well-formed permutation such as the expected count permutation, but the negative correlation required by theorem 5.2.9 should remain, along with the predicting properties.

7.3.2 Modifications with Partial Matching

When the fault-free counter weight is w, then k-Max DAPPER becomes $(k+w)$-Max DAPPER. This retains some of the advantages in terms of simulation reduction over conventional DAPPER, but results in variable test lengths. These arise from the fact that when less than a full length test is performed, the fault-free counter value is no longer w, but rather some value w_f, where $w_f \leq w$. The fault-free sequence must be matched with the modifier sequence for the n_f vectors. The analysis then proceeds as in section 7.3.1, with k_f set to

$$k_f = k + w - w_f \tag{7.3.6}$$

The expected test length is (for the independent error model, given k, n, and w):

$$E(n_f) = \frac{n(k+w)}{w + (n-2w)p_f} \tag{7.3.7}$$

which yields an expected value for k_f of:

$$E(k_f) = \left(1 - \frac{n}{w + (n-2w)p_f}\right)(k+w) \tag{7.3.8}$$

Unfortunately, the expected test length $E(n_f)$ is now related to the maximum test length n. While n_f may remain smaller than n, it will no longer reach and remain at a constant value. This will again result in a trade-off between matching ratio and simulation time, both for dictionary and demand-driven applications. The tradeoff will be partially offset by the fact that k_f will typically be greater than k, so a better estimate of detection probability can be obtained. These results will apply to the asymmetric model as well.

The probability that the last pattern observed was actually a failing pattern is (assuming the independent error model and discounting any deliberate perfect matching of part of the sequence):

$$P(\text{pattern } n_f \text{ is an error}) = \frac{k_f}{w_f + k_f}$$

If F patterns have been fixed at perfect matching, it is expected that Fp_f of the k_f errors will have occurred among these, giving:

$$P(\text{pattern } n_f,\ n_f > F, \text{ is an error}) = \frac{k_f - Fp_f}{w_f + k_f - Fp_f}$$

Clearly, as w increases with respect to k, the chances of the last pattern being an actual failing pattern decrease. The asymmetric model can cause some variation in actual values, but the trend remains.

7.3.3 Pros and Cons

Using k-Max DAPPER will reduce location time for random easy faults in demand-driven techniques, since post-test simulation will be performed on a smaller test length. Complexity is also reduced when a fault dictionary is to be constructed, as demonstrated in chapter 8. In addition, test time for easily detected faults will also be reduced, since the test may terminate after they have been detected k times. The information content of the signatures is improved, since the test length indicates the last failing quotient stream bit, in addition to the first indicated by the first-fail counter. This will aid in locating unmodelled faults.

There are some disadvantages to the method. Increases in test length will no longer result in better estimation of detection probability once a fault has been detected k times. This can be overcome by setting k sufficiently high as to give a reasonable accuracy. Another drawback is that faults with similar output behaviour may not behave differently over the first k errors, eliminating the chance to isolate them from one another. For example, if fault a *covers* fault b and $p_a = qp_b$, for some q, $0 < q < 1$, then after k detections of b, the chances of a and b having equivalent behaviour are q^k.* Again, setting k high enough can alleviate this problem.

The main benefit of k-Max DAPPER is that it provides a level of diagnostic resolution determined by the fault itself, not by any *a priori* scheme, while reducing the amount of simulation required for either demand-driven fault location or for dictionary construction (see chapter 8). Further, the absolute amount of simulation for a given fault will not increase with increasing test length n once n has reached the point where the fault has been detected k times. In chapter 9, results are reported for k equal to 10, 20, and 30.

7.4 Aliasing Probability

There are three types of aliasing which could apply to DAPPER: *cancellation error*, where errors cancel each other out in P/S conversion; *signature aliasing*, where a faulty and fault-free circuit have identical signatures; and finally *non-unique diagnosis*, where two distinct and non-equivalent faults have the same signature and hence cannot be resolved uniquely. These are investigated below:

7.4.1 Cancellation Error

Cancellation cannot occur in serial streaming, so Scan DAPPER is immune to it in the ideal case where $n_1 = n$. When P/S compaction is performed using a MINSR, however, or when the parity checking scheme of section 6.1.1.1 is used, this is not the case.

The chances of cancellation error in a MINSR are minimized by the delay in the register, since the outputs from successive test vectors are independent. Recall the circuit of figure 6.2. In a standard parity-tree checker neither a fault on line l, nor any other which propagated only through l could ever be detected. On the other hand, the MINSR in DAPPER can detect the fault when two successive vectors produce complementary values on line l. Although the lack of feedback may cause more errors to go undetected than in some conventional MISRs (an error may be cancelled by another single error at any time before it is finally shifted into the quotient stream, while in

* Fault a is said to *cover* fault b if every pattern which detects fault a also detects fault b. In conventional test generation, covered faults can be eliminated from consideration as part of a process called *fault dropping*. As an example, a stuck-at 1 fault on the input of an AND gate covers a stuck-at 1 fault on its output.

a type II register with feedback this is only possible until the error is shifted into a feedback tap), Bardell et al. ([Bar87], p. 131) show that such complete cancellation errors are "highly unlikely" In addition, algorithms 6.2 and 6.3 take cancellation into account when calculating detection probability, so these errors will not be considered further for aliasing

In the case of the parity checkers of section 6.1.1.1, any errors which appear on an even number of outputs spaced by $K_1 K_2$ bits will alias. Setting K_1 and K_2 greater than $\sqrt{m}$ eliminates this problem for two bit errors, but 4 bit errors at positions i, $i + aK_1$, $i + bK_2$, and $i + aK_1 + bK_2$ will alias, and larger sized errors combined from 4 bit errors can alias. Using two parity chains reduces the probability over a single chain (even of the same total length), at the cost of a more complex unit.* Any cancellation in the checkers does not necessarily preclude detection, since the LFSR signature is taken before the parity checkers are applied, but its occurrence could prevent the remaining signatures from assisting in the location of the fault In order to avoid this problem, and since the potential failing output set for each fault can be determined easily, it is important to select the test length for complete scanning (n_1) and the number of parity checkers ($K = K_1 + K_2$) carefully, and possibly to reorder the scan chain to reroute highly correlated outputs

7.4.2 Signature Aliasing

The second type of potential aliasing is a faulty signature equal to the fault-free value. When complete matching is used, the fault-free weight count is 0, detected errors can only increase the weight, and so the weight count cannot alias.

On the other hand, when a non-zero fault-free weight is permitted, aliasing within the weight counter becomes possible. The expressions of appendix A then apply, giving the probability of the weight counter aliasing, and hence a worst-case value for aliasing of all three signatures The results of [Rob88], however, suggest that weight counting and LFSR signatures tend to be orthogonal, in that their aliasing behaviour is roughly independent. If this is indeed the case, the aliasing probability of the LFSR signature (approximately 2^{-j}, where j is the LFSR length) may be multiplied by that of the weight counter to give a better estimate of total aliasing probability The first-fail counter on the other hand, is not independent of the weight counter in terms of its

* Exact expressions can be obtained, but they are based on assumptions about independence of output and probability of various sized errors occurring A detailed discussion is beyond the scope of this dissertation

aliasing, since placing restrictions on the position of the first failing pattern also affects the aliasing behaviour of a weight count (see section 7.4.3 for additional details).

As an example of the worst-case behaviour under the independent error model when a MINSR is used, consider a fault-free weight of 95 and a fault with detection probability of 0.274. The aliasing probability for a test length of 2048 is less than 10^{-100}. A fault with a detection probability of 0.001, on the other hand, has a weight count aliasing probability of 0.025. Results for various probabilities (from (A 2.5)) are given in Table 7.1. More complete analysis of the aliasing behaviour of weight tests under various error models is given in appendix A.

p_f	P_w	P_{nd}
0.2740	1.4e-140	1.6e-285
0.1000	6.28e-43	1.94e-94
0.0500	2.20e-21	2.39e-46
0.0200	2.13e-09	1.07e-18
0.0100	1.78e-05	1.15e-09
0.0050	0.0018	3.48e-05
0.0020	0.0148	0.0166
0.0010	0.0251	0.1289
0.0005	0.0169	0.3591

Table 7.1 Aliasing Behaviour, P_w^{ind}, $n = 2048, w = 95$

For small detection probabilities (on the order of $\frac{1}{n}$), the aliasing probability, while high, is dwarfed by the probability of not detecting the fault at all, which is for the independent error model

$$P_{nd}^{ind} = (1 - p_f)^n \tag{7.4.1}$$

In table 7.1, those entries with the highest aliasing probabilities also have non-trivial values for P_{nd}^{ind}. In fact, for $p_f = 0.0005$, there is a 36% chance of never detecting the fault. DAPPER's performance with with imperfect matching is improved by test lengths which are sufficiently long to detect the fault, and preferably to detect it several times.

7.4.3 Non-Unique Diagnosis

In the final case, non-unique diagnosis, all three signatures match for two unequivalent faults. First, consider two randomly-selected unrelated faults. If the faults have similar detection probabilities, the chances of their having the same count are good. Under the independent error model and assuming a MINSR (closed-form solutions for the generalized error model are difficult to obtain and in many cases not overly different from their independent counterparts), the probability of a fault with detection probability p_f having weight w_f equal to w with n test vectors is

$$P(w_f = w) = \binom{n}{w} p_f^w (1 - p_f)^{n-w} \qquad (7.4.2)$$

The chances of matching counts can be high, especially for small p_f and w. Given h faults, the chances of at least one fault, in addition to that observed, having count w are:

$$P_{nu,w} \leq 1 - \prod_{i=1}^{h} [1 - P(w_i = w)] \qquad (7.4.3)$$

Similarly, the probability of such a fault having first failing pattern r_f equal to r are given by:

$$P(r_f = r) = p_f \cdot (1 - p_f)^{r-1} \qquad (7.4.4)$$

which again can be high, especially for large p_f and small r. Given h faults, the chances of at least one fault, in addition to that observed, having first-fail r are:

$$P_{nu,r} \leq 1 - \prod_{i=1}^{h} [1 - P(r_i = r)] \qquad (7.4.5)$$

For aliasing to occur, both the first-fails and weight courts must alias, giving:

$$P_{nu,r\&w} \leq 1 - \prod_{i=1}^{h} [1 - P(w_i = w | r_i = r) P(r_i = r)] \qquad (7.4.6)$$

where

$$P(w_i = w | r_i = r) = \binom{n - r}{w - 1} p^{w-1} (1 - p)^{n-r-w+1} \qquad (7.4.7)$$

$P_{nu,r\&w}$ will tend to be large when there are many faults with similar detection probabilities and observed values are close to the expected values of these. In this case, it is the LFSR signature which must be relied upon to separate the signatures. This requires the knowledge of $P(s_f = s)$.

Lemma 7.3: The aliasing probability of an LFSR of length k bits tends to 2^{-k} as test length tends to infinity, provided that its input error probability is different from 0 or 1 infinitely often.

Proof: See [Wil86] (independent error model) and [Iva88b] (generalized error model)

Theorem 7.4: The probability of a fault f having LFSR signature s, where s is the signature of an unrelated fault a, tends to 2^{-k} as test length tends to infinity, provided that f can be detected in the presence of a

Proof: Let Λ be a circuit containing fault a, such that the signature of Λ in the presence of a is s. Let f be another fault, such that the detection probability of f in the presence of a is non-zero If f is inserted into Λ, then $P(s_f = s)$ is the aliasing probability of f when Λ with a is taken to be a fault-free circuit. By lemma 7 3, f's aliasing probability in Λ with a tends to 2^{-k} as test length tends to infinity.

Thus, theorem 7.4 states that

$$\lim_{n \to \infty} P(s_f = s) = 2^{-k}$$

where n is test length. as test length n tends to infinity So, the use of a sufficiently long LFSR can reduce the chances of non-unique diagnosis among unrelated faults to any desired level

When the faults are related, the problem is not so simple. Consider, for example, two faults where one covers the other, as discussed in section 7 3 3. In such cases, especially when detection probability is low, the only patterns which detected the covered fault may also have detected the covering faults. When the faults are detected at the same outputs, the output streams of the two faults are identical and thus indistinguishable by any method

The severity of the problem of related faults masking one another depends on both the number of such faults and the problems which will result from their misdiagnosis In many cases the faults are on the same gate, while occasionally they may be separated by a long path and consequently in physically disparate portions of the circuit. The solution to the problem appears to lie in test lengths that are sufficiently long to detect most faults more than once

7.5 Faults and Fault Models

It has been claimed that the Achilles' heel of signature-based fault location methods is the requirement that the fault model precisely predict faulty circuit behaviour so that the simulated and actual signatures match [Mer89] DAPPER is not entirely immune to this problem, so it is important that the fault model chosen accurately reflect circuit faults

Detection probability is the first criterion used by DAPPER in eliminating faults from consideration. Most previous work on obtaining detection probabilities (section

2.5.3) has used only the single stuck-at fault model, although some results have been reported for delay faults [Sav86], and multiple stuck-at faults [Cha86]. DAPPER is model-independent in that it may be used for any fault model which permits detection probabilities to be calculated for its fault set. This permits it to be used with the single stuck-at model, single delay fault model, and even a single bridging fault model. Statistical methods will allow detection probabilities to be estimated for virtually any other single fault model, including the layout-dependent inductive fault model of [She85]. It is stated in [She85] that using the inductive fault model can reduce or eliminate the need for multiple failure analysis. However, for more conventional models, the issue of multiple faults remains.

7.5.1 DAPPER with Multiple Faults

Although the reductions in location complexity obtainable with DAPPER are large, particularly with the k-Max method, they may still not be enough if the fault set is excessively large, as is likely to be the case if all multiple stuck-at faults are considered.* Since the number of potential multiple faults is astronomical, it is important to avoid ever having to enumerate them. Several approaches are possible.

The first might be to apply an additional hierarchical step and consider some faults as more likely to occur than others, thus, single faults could be analyzed first, then those of multiplicity 2, etc. The possibility of a demand-driven dictionary reduces the storage problems which could result from an exponential number of faults. However, once faults of a multiplicity beyond 2 or 3 are considered, the potential number of faults again becomes far too large for enumeration, so partial enumeration is unlikely to be an acceptable solution when faults of high multiplicity occur.

In order to avoid enumerating faults, it is necessary to identify some lines which are known to be fault-free [Abr80] [Raj87]. At this point, some exact knowledge of circuit output values is almost certainly required. In Scan DAPPER, this knowledge is available:

Given a first failing pattern r, it is known that a fault occurred on output $i = m - r \bmod m$ for test pattern $j = \frac{r}{m} + 1$, if r [illegible] m; otherwise its parity chain is known. Further, it is known that test patterns 1 through $j - 1$ were fault-free, and that outputs $i + 1$ through m were fault free on pattern j. This knowledge can be used by a call to a system such as CERBERUS [Raj87] to produce a list of known fault-free lines.

* Recall from section 2.3.3 that a circuit with L lines has potentially $3^L - 1$ multiple stuck-at faults compared with $2L$ single stuck-at faults.

The faulty output set can further reduce this list by eliminating faults which singly would have affected only outputs other than those where faults were observed

At this point, the set of potential failure sites will have been reduced The amount of this reduction will depend on the position of the first failing pattern — the later in the test sequence the better, and the number of outputs affected — the fewer the better. Faults could then be considered in order of multiplicity If the number of potential fault sites is reduced from L to L', the number of possible faults of multiplicity t decreases by a factor of q^t where $q = \frac{L'}{L}$ With large numbers of lines eliminated, the multiplicity of faults which can be feasibly analyzed can increase dramatically.

The heuristic methods of [Wai89] could be used instead of CERBERUS for identifying faulty lines These methods are likely to be faster, since they revolve around critical path tracing However, the multiple fault model used is not as general as that of [Raj87], allowing only a single fault effect per pattern (see section 4 2) Similarly, the effect-cause analysis of [Abr80] could also be used, although it is not suitable for delay faults and allows backtracking

The signatures of standard DAPPER and k-Max DAPPER can locate multiple faults in a similar fashion, but a simplifying assumption will have to be made This assumption is that the first error observed is the first which actually occurred, thus, no cancellation occurred in the MINSR. This assumption could potentially invalidate the results, but without it, an exponential number of cases would have to be considered Even with this assumption, a first failing pattern r could have resulted from a first error at output i on pattern $j = r - (m - i)$ for each i such that $j \geq 0$. In addition, no faulty output set information would be available

Clearly, multiple fault location is easier with Scan DAPPER than either Standard DAPPER or k-Max DAPPER. It is possible to combine Scan DAPPER with k-Max DAPPER and obtain the advantages of both — good location capability with multiple faults and reduced simulation effort Multiple fault location will be more difficult for faults with high detection probabilities If there are many such faults, a 2nd, or even 3rd failing pattern counter might be desirable for aiding location. In fact, it is demonstrated in chapter 8 that retaining all k failing patterns is a smaller signature overhead than the output blocks required by Intermediate Signature Collection (see chapter 4) and can provide better resolution.

7.5.2 Locating Unmodelled Faults

An additional problem can result when a fault produces output which is similar, but not identical, to the modelled fault. An example is an intermittent fault which is active a large part of the time. In such a case, the fault may be identifiable by analyzing the failing patterns, but not by simulating to a signature.

If an observed fault does not behave as any modelled fault, a search through the entire model may ensue if the p-value approach is used. This will have the effect of eliminating the first hierarchical reduction of the fault list, while the remainder will continue to apply. This may be eliminated by setting a threshold on p-values, below which a fault will be declared unmodelled. This threshold must be set sufficiently low so as to make the missing of modelled faults unlikely.

DAPPER is not utterly defenseless when an unmodelled fault occurs. The hierarchical nature of the diagnosis procedure permits some information about the fault to be deduced. For example, an estimate of its detection probability is available, indicating whether the fault is easily detectable, or whether it is random-pattern resistant and hence potentially more subtle in its effects. Further, the position of the first error detected is available. In serial streaming the exact circuit output and input vector are known, and even in P/S compaction the error position is restricted to at most m output/vector combinations. An offline analysis, using standard deterministic techniques (e.g. path sensitization, critical-path tracing, etc.) could provide a set of potential trouble spots. The methods used to locate multiple faults could provide some idea as to the nature of the fault, although they could be misleading for an intermittent failure (see section 2.3.3). If k-max DAPPER is used, many faults will also have a last failing pattern available, which will provide additional information to isolate the fault(s). The loss of information over a list of all failing patterns will lower resolution, but some identification could still be made -- something virtually impossible without an example of a failing pattern.

Chapter 8 Performance of Hierarchical Fault Diagnosis

The previous chapter introduced several implementations of hierarchical fault diagnosis, collectively known as DAPPER. In this chapter, the performance of these methods is compared with that of other methods previously described in the literature (see chapter 4) over a range of situations These situations roughly encompass the potential applications of fault location in circuits tested with random or pseudo-random vectors and are divided into three broad categories:

1 Dictionary constructed in advance.

2 No dictionary constructed.

3. Dictionary constructed as required (demand-driven).

When necessary, further distinctions are analyzed within each of these categories depending on the fault model being used, whether test hardware is external or built-in, and whether or not the circuit is a scan design.

Two methods from chapter 4 have been selected. signature by simulation, and intermediate signature collection Algorithmic analysis has not been sufficiently developed for any useful comparisons to be made Signature by simulation will be considered only in situation 1, where the dictionary is constructed in advance, since it is of no use whatsoever without a dictionary. Similarly, the benefits of intermediate signature collection are limited if a dictionary is produced in advance, so it will only be considered in situations 2 and 3

In all cases the following symbols will be used·

- Number of test vectors. n
- Number of circuit outputs. m
- Number of circuit gates· G
- LFSR length (if used) j

Additional symbols will be defined when they are first employed.

The methods are compared with respect to a variety of cost and performance criteria These are briefly summarized below·

- **Completeness.** The probability that the method is able to provide a unique signature for each fault.
- **Aliasing.** The probability that a randomly-selected fault detected by the test set will go undetected after compaction into the location signature
- **Start-up cost.** This includes the cost of producing the dictionary (if one exists), which is divided into two categories: fault simulation and output compaction Each of these costs is reported in terms of time and space complexity For methods which do not use a dictionary, start-up costs cover any preprocessing which may be necessary.
- **Run-time cost.** For dictionary methods, this cost includes the storage cost for the fault dictionary and the time required to locate a given observation within the dictionary. For demand-driven location methods, the per-diagnosis cost is given, in terms of computational and space complexity
- **Observation cost.** This cost reflects the cost of obtaining the signatures from the circuit. A time factor is involved, since scan techniques cannot in most cases test a circuit at operating speed. Test overhead is measured in three ways First, the number of test vectors required, second, number of ROM bits required to perform the data compaction, and third, the number of register (RAM) bits required to observe the results. The last two categories reflect hardware cost, while the first reflects the time needed to apply the test, and hence the throughput of the tester
- **Information content.** This reflects the information inherent in the signature which can be used for further diagnosis in the event that no modelled fault is found to be responsible for the observed behaviour, possibly permitting some kind of "effect-cause" analysis [Abr80].

Each of these factors is important, although their relative importance will depend on local circumstances.

8.1 Completeness of Methods

Because all of the methods investigated involve data compaction none can guarantee with absolute certainty that no two faults with distinct fault effects will have the same signature Nonetheless, a wide variation is possible in the probability of this occurrence

For example, in signature analysis schemes each of the possible signatures tends to be equally likely (Theorem 7.4, [Wil86], [iva88b]), so the chances of a given fault having a given signature are approximately 2^{-j} where j is the length of the signature register. With h different faults, the probability that each will have a different signature (assuming $h \ll 2^j$) becomes:

$$P_u = P(\text{unique signatures}) = (1 - 2^{-j})^{0.5h(h-1)} \tag{8.1.1}$$

With 2000 faults and a 16-bit register, P_u is virtually 0 — there is less than 1 chance in 10^{13} that all the signatures will be unique. If j is increased to 24, there is an 89% chance that the signatures will be unique. Finally, with a 32-bit register, there is a 99.95% certainty that the signatures will be unique. Figure 8.1 shows how P_u varies with different values for h and j.

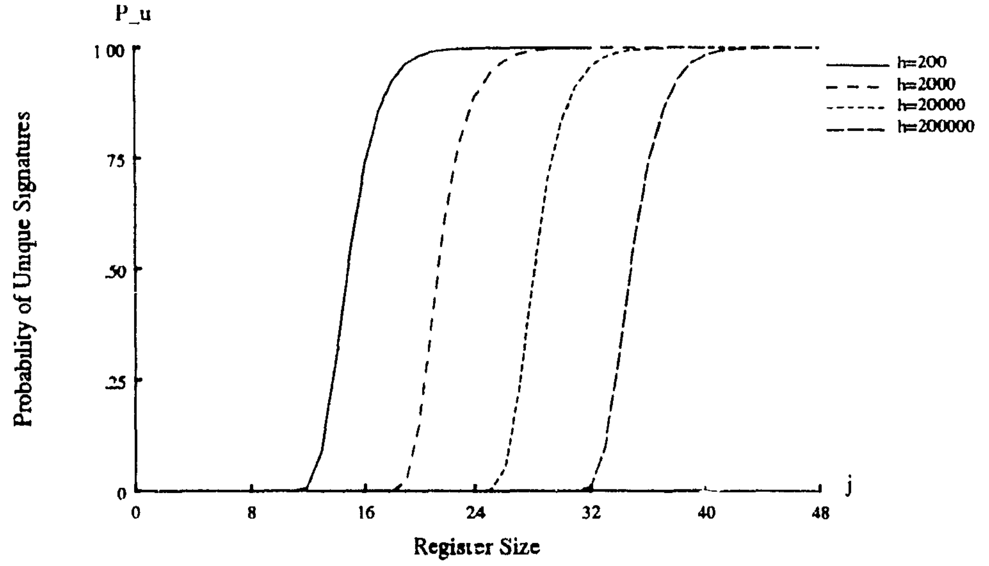


Figure 8.1 Probability of matching signatures versus register size

Figure 8.1 shows that the number of faults to be distinguished has a strong effect on the required register length. The hierarchical DAPPER techniques are able to use a smaller LFSR, since large numbers of faults are eliminated through other signatures. For each tenfold increase in the number of faults to be considered about 8 additional register bits are required

Note that the above does not discuss related faults (see section 7.3.3). The case where one fault covers another can have a significant effect on the chances of signatures having equivalent values, particularly for weight counts [Ait88d] It also increases the chances that the test output sequences will be equivalent, thereby precluding distinction between the two faults

In conclusion, all compaction-based methods require sufficiently long signatures before it can be stated that the compaction is unlikely to be reducing potential diagnostic resolution significantly.

8.2 Aliasing of Methods

Aliasing has been covered in previous chapters for both the DAPPER schemes and signature analysis methods. In each case, having signature registers which are long enough to ensure the completeness of the methods will also reduce aliasing probability to an insignificant value. Only one method, however, is able to reduce aliasing to 0, that being the ideal Scan-DAPPER method of chapter 7 with n_1 set to n. Of course this method requires a complete record of the fault-free output, which is likely to be unacceptable overhead in most cases. The other DAPPER methods have greatly reduced overhead with very low aliasing probability. For instance, Scan DAPPER with $n_1 < n$ and Standard DAPPER contain only cancellation aliasing within the parity unit and MINSR respectively. MISR compaction, on the other hand, contains both cancellation and signature aliasing (see section 7.4). Cancellation aliasing in shift registers is believed to be much less likely than signature (matching remainder) aliasing [Bar87].

8.3 Dictionary Constructed in Advance

In this section it is assumed that a full fault dictionary is to be constructed before any fault location is performed. With signature by simulation techniques, the key to this dictionary will be a MISR or LFSR signature, while for the DAPPER methods several keys will be used, permitting a multi-step location process, if so desired.

The most significant costs in any dictionary system are those of creating and storing the dictionary. For example, storage costs prevent the accumulation of a dictionary listing each failing pattern for each fault. A circuit with a hundred thousand gates and a scan chain of say a thousand elements to be tested with a few thousand random patterns would require a dictionary hundreds of gigabytes in size, even for a fault model as simple as the single stuck-at. Even a deterministic set of only a few hundred vectors would not reduce the dictionary size to manageable proportions. On the other hand, a signature on the order of 100 bits in size would allow the dictionary to be stored in a few megabytes — still large, but well within the capabilities of contemporary storage devices.

Thus, for large circuits, only data compaction methods such as signature analysis, counting methods, and first-fail techniques permit the storage of fault dictionaries. While these methods all result in dictionaries of approximately the same size, the costs of developing them vary dramatically.

8.3.1 Dictionary Development Costs

Only the time complexity of dictionary development will be considered in this section, since the space complexity is identical among all the methods investigated.

Recall from chapter 2 the fault simulation cost assumption (assumption 2.1), which states that if the computational effort required to simulate a circuit for one input vector and one fault in order to produce one output vector is x,* then the simulation effort required to produce n output vectors for one fault is nx and the effort required to produce one output vector for each of h faults is hx. So, given x, the cost of simulating each of 1000 faults for each of 2000 patterns is 2 000 000x. This full effort is required if the data compaction method requires each output bit for each fault. Signature analysis and the other signature techniques of chapter 3 fall into this category. If the first failing pattern is used instead, then the effort diminishes, because the fault need not be simulated once it has been detected. This effect is demonstrated in figure 8.2.

The portion of the graph above the first-fail line represents the relative effort in compiling a first-fail dictionary (as well as cumulative fault coverage), as compared with the total area of the graph, which represents the effort required to calculate a full simulation dictionary. In the case of figure 8.2, this is about 7.5% of the total effort, or about 150 000x. Increasing the test length to 4000 to cover some of the more difficult faults would lower the relative effort to 4.3% (172 000x) for the first fail technique. Savings can be determined directly from fault coverage curves, or may be estimated using the arctangent model described in section 5.4.1. However, precise values cannot be obtained without actually constructing the dictionary.

Of course, a single failing pattern is unlikely to provide the level of resolution that could be obtained when multiple fails are observed. For this reason, the kth failing pattern may be used instead. In this case, the simulation effort is increased, although

* The computational complexity of performing a full simulation for one fault and one vector is $O(G^2)$ where G is the number of circuit gates or $O(L)$ where L is the number of circuit lines. The space complexity is $O(L + G)$. The complexity of compacting this output into a signature is $O(m)$, where m is the number of circuit outputs. This value is essentially independent of the compaction technique used. See chapter 2 for more details.

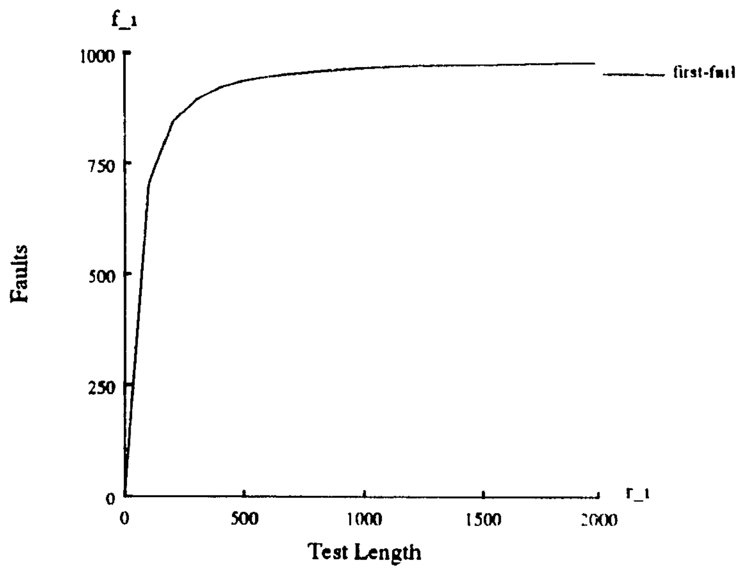


Figure 8.2 Relative simulation effort with first-fail

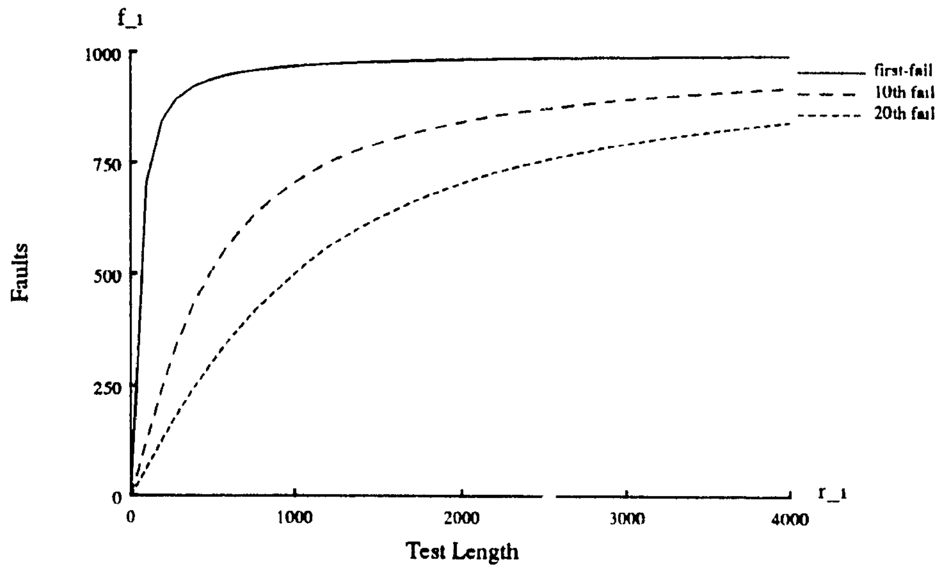


Figure 8.3 Relative simulation effort with first-fail

the actual amount of the increase will depend on k. This effect is demonstrated in figure 8.3.

While the first failing pattern curve corresponds to 7.5% of the full simulation effort, the 10th fail requires about 38% and the 20th fail 52% of the entire effort. Again, if the test length is increased to 4000, the relative effort decreases to 25% in the case of 10th fail, and 38% for 20th fail. Thus, the kth fail measures share the property of decreasing relative cost with increasing test length.

Another way of looking at these cost figures is to compare absolute simulation costs. Doubling the test length will double the absolute simulation effort for full simulation signatures, while increasing that for *k*th fail techniques by substantially less For the example above, doubling test length from 2000 to 4000 results in 15% more work for a first fail dictionary, 32% more for a 10th fail and 46% more for a 20th fail.

Such reductions are important when investigating the tradeoffs between fault coverage and dictionary cost. While doubling the test length might increase fault coverage from 99% to 99.5%, a 100% increase in dictionary generation cost might be too high a price to pay If the generation cost was to increase only 20%, the improvement in quality might be cost-effective. These savings will also be realized if the test set is changed during the circuit's lifetime.

The percentage reductions observed above are similar to those obtained experimentally for the ISCAS-85 benchmark circuits [Brg85]. The results are reported fully in chapter 9, but a brief summary is shown in table 8.1.

Circuit	Full Sim.	10th fail	20th fail	30th fail
alu	1.0	0.05	0.10	0.13
C432	1.0	0.10	0.17	0.22
C499	1.0	0.15	0.21	0.26
C880	1.0	0.14	0.22	0.27
C1355	1 0	0.22	0.28	0.34
C1908	1.0	0.28	0.33	0.37
C2670	1.0	0.27	0.32	0.36
C3540	1.0	0.18	0.25	0.31
C5315	1.0	0.12	0.19	0.25
C6288	1 0	0.03	0.04	0.06
C7552	1 0	0 19	0.26	0 32

Table 8.1 Relative simulation effort, $n = 2048$

In the best case (circuit C6288) a 10th fail dictionary requires only 3% of the effort of a full dictionary for a test length of 2048 For less randomly testable circuits. the effort goes as high as 28% for a 10th fail with circuit C1908. or 37% for a 30th fail for the same circuit In all cases a substantial savings is observed. even for the short test length examined Longer test lengths yield greater reductions For example, increasing the test for C880 to 21568 vectors (a complete test set) yields results of 3 7% for the 10th fail. 5 5% for 20th fail. and 6 6% of a full dictionary effort for the 30th fail test.

Thus, constructing a 20th fail dictionary for k-Max DAPPER ($k = 20$), requires about one twentieth of the cost of generating a dictionary for a standard signature technique. With testing effort taxing the abilities of computer systems, such savings can make the difference between feasible and infeasible effort.

Note that it is the redundant faults which contribute most of the effort in dictionary construction. For example, a complete test for circuit C3540 (maximum length 50 016 vectors) requires 5.4% of the full signature by simulation dictionary effort to construct a 20th fail dictionary. If the 137 redundant faults were identified in advance, this is reduced to 1.5% of the full effort. Potential savings like these show that a quick pass through the circuit with a deterministic redundancy identifier like Socrates [Sch88] may be justified for long test lengths. The reduction of 4% of overall effort may make little difference for a signature by simulation dictionary, but it represents a threefold savings for the 20th fail DAPPER equivalent.

It is possible to reduce the overhead of standard signature dictionary techniques by taking signatures at different times (after say 1000 vectors and then again at the end of the test). The disadvantage of such methods when compared with the kth fail techniques is their lack of provable resolution. A fault recorded by the first signature may have been detected by only one test pattern, making location conclusions based on it much less reliable than had it been detected numerous times. With the kth fail signatures, a fault is guaranteed to be detected either k times, or else the maximum possible with the test set. The fact that the signature is controlled by the fault permits this guaranteed resolution.

8.3.2 Dictionary Storage Costs

As mentioned previously, a complete dictionary listing all failing patterns for all faults is infeasible for VLSI circuits, even with simple fault models. Some form of data compaction is required. The length of signatures then becomes an important factor in the size of the fault dictionary constructed. Table 8.2 lists the sizes for a variety of techniques. Note that a kth fail signature consists of either the pattern where the kth fail was observed, which will have a value in the range of k to n, or the counter value after n vectors, which will be less than k.

Table 8.2 shows that the signatures are similar in size. Since $\log n$ is unlikely to exceed say 25, the fail indicators and weight counters will in most cases be the smallest signatures. LFSR length will depend on the resolution desired (see section 8.1), but will typically be less than about 50 bits. The longest signatures are MISR and failing

Signature	Size
first fail indicator	$\lceil \log n \rceil$
kth fail indicator	$\lceil \log n \rceil$
weight counter	$\lceil \log n \rceil$
LFSR	j
MISR	m
failing output reg.	m

Table 8.2 Sizes of various signatures

output measurements, whose lengths depend on the number of circuit outputs and could become quite large.

Suppose, for example, that a circuit with 200 outputs and about 20 000 faults is tested with 100 000 random patterns. A MISR signature by simulation scheme will have entries of 200 bits, an LFSR signature by simulation will require about 40 bits to ensure coverage, and a k-max DAPPER scheme would require 17 bits for each fail indicator and perhaps 16 bits for the LFSR for a total of 50 bits. The complete dictionary sizes are 400K, 80K, and 100K bytes respectively. Variations in any of the circuit parameters will cause corresponding changes in the signature sizes. In general, the MISR signature will be the longest of the three, but differences in the other two will depend on test length and number of faults

8.3.3 Dictionary Look-Up Costs

Once a dictionary has been created, efficient means can be developed to access it. These may be developed independently of the signatures or in conjunction with them. Only in the latter case can differences between techniques be examined in detail. A look-up table approach can be used when the number of possible values is small — In most cases, LFSR or MISR signatures will be too long for such techniques, but first-fails or weight counts could employ them if test length was less than say 100K patterns. Otherwise, binary search or hashing methods could be used In general, the effort will be similar regardless of the diagnosis or dictionary search method used

8.3.4 Test Application Costs

Test application costs encompass both test application time and tester hardware costs. These latter costs involve the costs of vector generation (stored pattern or pseudorandomly generated), signature storage costs, timing features, etc. Many of these factors are highly dependent on the individual process and beyond the scope of this dissertation, but comparisons may be made on the basis of test application time, and the amount of storage (ROM and RAM) required by the different compaction methods.

The reductions in dictionary development time for the DAPPER methods over signature by simulation techniques demonstrated in section 8.3.1 also apply to test application costs. Fewer vectors are required (on average) for each signature, and there is no need to apply those which are not used. So, given equal probability of any fault occurring, the simulation savings achieved by DAPPER translate into reduced average test time for faulty chips (neglecting the time required to set up the chips on the tester). Of course, the worst-case test length remains as n, the length always incurred by the signature by simulation methods. In all methods, fault-free circuits will have test length n, so the process yield will affect results. For low yields, faulty circuit test times will dominate and DAPPER should perform significantly better than signature by simulation, while for high yields, fault-free circuits will dominate and the test lengths will be virtually equal.

Thus, the figures in table 8.1 represent a lower bound on the relative test length expected using k-Max DAPPER as opposed to a full length signature scheme for $n = 2048$. The accuracy of such projections is limited, however, since it relies on the assumption that every fault has an equal probability of occurrence. The kth fail techniques will have shorter average test length, but whether the relative times are more, less or the same as those of table 8.1 will depend on whether the faults which actually occur tend to be less, more, or equally detectable on average as the modelled distribution.

The reductions in test time are not without cost, however. The DAPPER methods require the storage of a fault-free sequence (n bits with a MINSR, $mn_1 + K(n - n_1)$ bits for Scan DAPPER). This read-only (ROM) storage is required only for the fault-free signature for MISR (m bits) or LFSR-based (≤ 50 bits) signature analysis. If the DAPPER overhead is excessive, as is possible with an on-chip unit, the methods of section 6.2 may be used to reduce it. Nonetheless, the storage requirements of the testing unit for the kth fail methods are more complex than those of signature analysis techniques, although they are still simpler than needed for deterministic testing.

The remaining test circuitry is similar for all techniques. The number of writable signature register bits was discussed in section 8.3.2. Scan-based methods such as LFSR signature analysis and Scan DAPPER require scan hardware, and DAPPER without SCAN requires a MINSR, which is slightly less complex than a MISR (no feedback)

Thus, the only significant hardware difference between the methods is the fault-free sequence storage. Whether this cost will preclude taking advantages of DAPPER's dramatic reductions in dictionary construction cost and test application time will depend on individual circumstances.

8.3.5 Information Content

One problem with signature analysis is that fault location via a dictionary is essentially a binary process — either the fault is present or it is not. If no dictionary entry matches the observed signature, no information whatsoever is available as to the nature of the fault. The hierarchical signatures employed by the DAPPER methods are able to reduce the negative effects of this situation, since each signature contains useful information about any fault which may have occurred. The weight count provides an estimate of detection probability. The first-fail counter provides an actual failing pattern and output in the case of Scan-DAPPER, or a set of m possible pattern/output combinations in the case of a MINSR. Finally, if present, a failing output register provides information on the paths the fault can take. This information partially characterizes the fault and can aid tools such as those of [Abr80] or [Raj87] in eliminating fault-free regions of the circuit from consideration. Of course, the use of improved fault models, such as those of [She85], can greatly increase the effectiveness of a fault dictionary.

8.3.6 Effect of Partial Matching

Recall that in section 6.2 partial matching of the quotient sequence was suggested as a means of reducing the hardware overhead penalty of the fault-free sequence generator. When this occurs, kth fail techniques for small k are less useful, since many (or all) of the k errors may be due to imperfect matching. In section 7.3.2 $k + w$-Max DAPPER was suggested as an alternative, with w being the expected weight count for the fault-free circuit. Since w is dependent on test length, increased maximum test lengths will increase each individual test length as well.

For example, suppose that a given fault (say α) is detected about once every 20 vectors. A 10th fail signature will be expected to take about 200 vectors to tally. Increasing the maximum test length beyond 200 in 10-Max DAPPER will have no effect on fault α's test length. On the other hand, suppose that $10 + w$-Max DAPPER is to be used and the matching rate is about 90%. For a maximum test length of 200, w is expected to be 20, and the final count in the presence of α is expected to be 29, below the value of 30 which would have terminated the test. Increasing the maximum test length to 2000 will increase the expected weight to 200, meaning that 210 errors must now be counted before test termination. This is expected to occur at vector 1500 Similarly, increasing the length to 10000 increases the expected test length to 7211

Thus, the savings in dictionary construction costs and average test time are unlikely to be as dramatic with $k + w$-Max DAPPER as they are for k-Max DAPPER Savings are likely to be only on the order of 25% to 50%, as opposed to the order of magnitude reductions observed with k-Max DAPPER. Nonetheless, the signatures retain much of their information content, particularly if the first few bits are fixed at 100% matching in order to allow a meaningful first-fail value. The tradeoffs between savings in hardware and additional dictionary costs should be considered before choosing a particular implementation of DAPPER.

8.4 Demand-Driven Diagnosis, No Dictionary

This section compares the performance of fault location methods when no dictionary is to be constructed. Diagnosis will proceed by generating the appropriate sections of the dictionary as needed, and these sections will not be retained after the diagnosis is complete The two methods to be compared are DAPPER and Intermediate Signature Collection (ISC), since, as was mentioned earlier, the signature by simulation methods cannot be applied in the absence of a dictionary

As opposed to dictionary systems, where the start-up cost of creating the dictionary is the most significant, the major costs of demand-driven systems are the run-time costs of conducting the diagnosis. These include test time, tester complexity, and in the case of the two methods analyzed in this section, the amount of post-test simulation necessary to resolve faults.

In addition to costs, quality considerations must also be addressed, since results must be reliable and accurate. Examples of quality criteria are guarantees of correct diagnosis, size of fault classes, extensibility of the methods, and information content of the signatures in the event of unmodelled faults

8.4.1 Start-up Costs

Since there is no dictionary to be generated, the start-up costs of the demand-driven techniques are greatly reduced over those of section 8.3.1 This reduction in cost effectively removes them from the critical path and allows diagnosis to commence with production. Any fault detection start-up costs, such as test set evaluation, will not be considered, since they will be the same for both methods.

In ISC, the only start-up cost is the generation of the fault-free MISR signatures. Each of these involves a fault-free circuit simulation over a set of L (usually 256) vectors. Circuit outputs are separated into actual primary outputs and those associated with scan cells. Let the number of primary outputs be m', and the total number be m. The time complexity of the circuit simulation for each vector is $O(G^2)$ and the complexity of each signature generation is $O(m)$, for a total time complexity of $O(n(G^2 + m))$. The space complexity of the operation is $O(G^2 + m')$. The storage requirement for the resulting signatures is $m'nL^{-1}$ bits.

As with ISC, fault-free signatures must be generated before DAPPER can be used. In addition, however, detection probabilities must be obtained for each fault. The cost of these will depend on the algorithm used. For example, if COP [Brg84] is used, the cost will be small time complexity $O(G^2)$ for each fault, space complexity $O(G^2)$ total. At the opposite end of the spectrum, if statistical sampling is used with say l vectors, the time complexity will be $O(lG^2)$ for each fault. Other algorithms will lie somewhere in between (not counting the exhaustive approaches with their worst-case exponential complexity). Section 2.5.3 provides more information about testability measures. As is demonstrated in chapter 9, the more accurate the detection probability information, the better the performance of DAPPER. Once the detection probabilities have been calculated, they must be sorted to permit the construction of a diagnosability profile (see chapter 7). The time complexity of this sort operation is $O(h \log h)$ where h is the number of faults, and its space complexity depends on the algorithm used ($O(h)$ for an in-place sort).

The complexity of calculating the fault-free DAPPER signatures is identical to that for ISC. The storage cost of these signatures will depend on the approach chosen. Standard and k-Max DAPPER require n bits for the fault-free MINSR sequence, plus $2\log n$ bits for the counts and another j for the LFSR signature. The relationship between this value and the $m'nL^{-1}$ will depend on the relative values of m' and L, but in many cases, especially for larger circuits, will be comparable. In Scan DAPPER, the storage cost depends on the selection of n_1 and K (the number of vectors to be scanned out whole and the size of the parity compressor for the remainder), and will be $mn_1 + K(n - n_1)$ bits. In virtually all cases this will be larger than for ISC, although typically by only a constant factor of about K.

In conclusion, start-up costs for DAPPER are somewhat higher than for ISC, but as this is only a one-time cost, its effect will be minimal if DAPPER is able to outperform ISC in run-time cost.

8.4.2 Run-time Costs

The run-time cost of fault location is the largest in demand-driven fault diagnosis systems. In comparing these costs, two situations will be investigated In the first, faults which occur will be from a single fault model, such as the single stuck-at, stuck-open, bridging, or even the layout-dependent inductive fault model of [She85] The second situation will allow faults from multiple fault models to occur.

8.4.2.1 Single Fault Models

If the diagnosis method is to resolve only single faults (when realistic faults are included, such single models are very powerful [She85]), the output storage area of ISC becomes unnecessary and analysis can proceed by simulating predicted faults and comparing the results to those observed The major contrast between ISC and DAPPER in this situation is that DAPPER attempts to reduce the number of faults being considered while ISC limits the test length under investigation.

In this situation, ISC will resolve faults by simulating each one over the L vector period where a fault is first detected by the MISR Faults which match the MISR signature will be retained, all others will be dropped. The process is repeated for other failing intervals until either the fault list is reduced to a single fault,* or the potential fault list has remained unchanged for a few intervals, in which case it may be that these faults form an equivalence class. The question of how many intervals must be simulated in this instance is a difficult one Identifying non-trivial equivalent faults is an NP-complete problem, so a sufficient number of errors should be observed before claiming equivalence. On the other hand, investigating additional failing blocks increases both test time and simulation cost. Using assumption 2 1 which allows the comparison of simulation effort in terms of the number of fault-patterns simulated the lower bound for simulation is Lh fault-patterns, where h is the number of faults in the circuit

In DAPPER, the error weight count is used as the first step in eliminating potential faults, followed by the first failing pattern, using the method described in chapter 7 If the h initial faults are reduced to h_1 by detection probability and h_2 by the first failing pattern, and if the mean value for the first failing pattern is r_m, then a typical amount

* The issue of an unmodelled fault occurring will be considered later

of simulation to locate a given fault will be $r_{av}(h_1 + kh_2)$ for k-Max DAPPER. This will be less than the simulation involved in ISC whenever

$$r_{av}(h_1 + kh_2) < Lh \qquad (8.4.1)$$

Consider the following example of benchmark circuit C880 (complete results are given in chapter 9): Observed values are 0 09h for h_1, 1.15 for h_2 There are 942 modelled faults, and using 256 for L, and 20 for k this yields a maximum value for r_{av} of:

$$r_{av} < 2228$$

The observed value for r_{av} is 111. For tests where r_{av} is less than the maximum value from (8 4 1), DAPPER will outperform ISC by an amount equal to the ratio of the two sides of (8 4 1) In the example above, DAPPER will outperform ISC on average by a factor of 20

Decreasing h_1, h_2, or k tends to raise this maximum value, as does increasing L. Experiments reported in chapter 9 demonstrate this improved performance

Thus, DAPPER is able to offer significant cost reductions over ISC for countable fault models in terms of the amount of post-test simulation required to make a given diagnosis. In addition, resolution is guaranteed by the fault itself, so the tester may terminate the test after the fault has been detected either k times, or as many as permitted by the entire test set, whichever is lower. In the case of ISC, however, the tester cannot make this decision without external information unless k failing intervals are observed in most cases many more than the "few" advocated in [Wai87] and [Wai89].

The additional failing blocks which must be observed suggest that for comparable guaranteed resolution, average test lengths for faulty circuits and thus tester time will be longer with ISC than with DAPPER, although the differences will not be so pronounced as those observed with signature by simulation techniques, and, of course, fault-free circuits still require the maximum test length

In addition, tester costs may vary The storage cost of the various fault-free values is comparable, as indicated in section 8 4 1. ISC requires an m' bit MISR as its only writable memory (RAM), while DAPPER requires 2 log n bit counters Scan DAPPER requires an extra $m + K$ bit register plus slightly longer counters (log($mn_1 + K(n - n_1)$) bits), a K_1 and K_2 bit parity registers, demultiplexers and the assorted clocking hardware Thus, DAPPER's tester will be somewhat more complex, but the improved performance and resolution should be ample to offset the cost.

8.4.2.2 Multiple Fault Models

When a multiple fault model is used, fault enumeration is no longer possible, so, in addition to signatures, actual output vectors are required for diagnosis. A variety of methods of locating failures given faulty and fault-free circuit responses have been proposed [Abr80] [Raj87] [Wai89] (see section 2.3.3 for more information). Any of these techniques may be used with faulty output information generated by either DAPPER or ISC, so the cost of using them will be constant. Instead, the costs of obtaining the vectors, as well as vector quality, needs to be addressed. The two methods compared in this section are ISC as described in [Wai89] (section 4.2) and the k-Max Scan DAPPER technique (section 7.5.1). The latter is the DAPPER application most suited to the externally-tested full-scan circuits for which ISC works.

The general principle employed by both ISC and DAPPER in developing signatures is that error streams with random input vectors tend to be sparse — the vectors which detect faults are often vastly outnumbered by those which don't. ISC takes advantage of this behaviour by recording blocks of L output vectors which contain an error at some point. DAPPER, on the other hand, records the positions of some individual error patterns, and with the failing output register gives a global view of where errors have been observed. This latter process takes advantage of another common property of error streams — errors tend to be restricted to a few outputs.*

The first failing pattern recorded by DAPPER not only indicates an error, but it also implicitly states that all previous patterns were fault-free. Additional failing patterns can be recorded for the cost of a register long enough to record a pattern number. (In the case of hybrid Scan DAPPER, this is $\log(mn_1 + K(n - n_1))$, no more than 30 bits, given m on the order of 1000, n_1 at most a few thousand, n under 1 000 000, and K around 20.) The number of these registers can be balanced against the desired resolution and acceptable tester cost.

Those outputs which are stated in the failing output register to have never failed are known to be fault free for all input patterns present in the first n_1 patterns. After that, the failing output register records errors found by the parity checkers. Again, tradeoffs in the value of n_1, the number of parity checkers and the ordering of the scan chain should be able to resolve any problems with cancellation aliasing. This flexibility is not shared by ISC.

* The exception to this is catastrophic failures such as scratches across the silicon which could potentially affect all outputs. Of course, locating every fault site in such a situation is probably unnecessary.

Intermediate signature collection is much coarser in its resolution than DAPPER. Blocks containing failures are recorded in their entirety, (mL bits). Blocks which passed are known implicitly to be fault-free (disregarding the possibility of aliasing in the MISR, which may or may not be significant). The blocking of errors tends to prevent the kinds of guaranteed coverage possible with DAPPER, since 3 failing blocks could contain 3 erroneous patterns, or $3L$. Multiple failing blocks must be observed for accuracy, since a single error is insufficient for reliable diagnosis. To guarantee that k errors have been observed whenever possible will require that testing continue until k failing blocks are recorded. The storage overhead in this case is likely to be unacceptable.

In terms of tester overhead when multiple fault models are to be used, DAPPER tends to use more ROM while ISC requires more RAM. For example, the total ROM storage of ISC is $m'nL^{-1}$ bits, while for DAPPER it is $N = mn_1 + K(n - n_1)$. Suppose that m is 500, m' is 50, K is 20, L is 256, n is 50 000 and n_1 is 250. This results in 9766 bits of ROM for ISC. On the other hand, the DAPPER tester requires 1.2M bits of ROM — a much larger value. However, in terms of RAM the situation is reversed. To record three failing blocks, ISC requires 384K of RAM, while DAPPER, assuming $k = 30$ failing patterns are to be recorded in total, requires 520 bits for a failing output register, and $\log N = 21$ bits for the weight counter and each failing pattern indicator. Even if each of the 30 patterns are recorded, that still totals only 1171 bits of RAM.

The cost differences between RAM and ROM can easily be summarized. In the previous example, each test with ISC will produce 384K of data, which may either be stored in the tester while the next chip is diagnosed, or downloaded to a computer for immediate analysis. With DAPPER this amount is just 1171 bits. If 4 or more chips are to be tested before downloading commences, total tester memory (RAM plus ROM) will be greater for ISC than for DAPPER. In either case, the communication overhead between the tester and the diagnosis computer will be 300 times greater with ISC than with DAPPER, and the ISC data may contain as few as three failing patterns. Thus, it appears that the static storage cost of ROM will be much less significant than the dynamic storage cost of RAM.

In conclusion, when a multiple fault model is used, DAPPER is able to outperform ISC in terms of test quality, test time, and tester overhead, given reasonable assumptions about each of these costs.

8.4.3 Information Content

With any fault location method, it may be impossible to obtain an accurate diagnosis. The problem is particularly acute with signature schemes, as was noted in section 8.3.5. Both ISC and DAPPER are equipped to provide some information about unmodelled faults which may have occurred. ISC will give sets of L patterns in which an error is located, while DAPPER will indicate failing patterns and approximate detection probability

In the context of single fault models, DAPPER's information is likely to be more useful, since ISC in this case does not store actual output values, and the previously discussed limitations of signature analysis will come into play In DAPPER, the attempt to diagnose an unmodelled fault could, if no other termination conditions were included, result in the generation of a near-complete first-fail dictionary. If such were the case, the results should be stored with the diagnosability profile to reduce the costs of its recurrence. Alternatively, a first-fail dictionary could be assembled and inserted into the diagnosability profile before diagnosis commenced, further reducing the run-time costs. Generating this dictionary is required in any case if the fault coverage of the test set is to be determined

With multiple faults, resolution problems tend to result from the algorithm used In many cases, increasing the amount of fault information (e.g increasing k for DAPPER or the number of failing blocks for ISC) will allow diagnosis, while in others diagnosis is impossible. In these cases, the diagnosis algorithms will often give a listing of potential faults sites Because DAPPER is able to provide fault information in a much more compact form than ISC, it is likely to be able to give better information to the deterministic diagnosis algorithms.

8.4.4 Extensibility of Methods

Intermediate Signature Collection is designed to work only with scan-design circuits The method cannot be extended to partial scan or non-scan sequential circuits, since there is no way to reset the circuit state effectively after each block of L vectors DAPPER, on the other hand, can be extended to sequential circuits and some preliminary results have been reported [Ait89c] The results so far indicate that generating suitable random test sets for sequential circuits is a much more difficult problem than diagnosing the resulting failures, at least in the context of single fault models Nonetheless, the potential to diagnose failures in sequential circuits with DAPPER is important, especially in the case of features such as partial scan.

8.5 Demand-Driven Dictionary Construction

It is possible to store pertinent results from demand-driven diagnosis once they have been obtained so that over time a dictionary will be constructed. This demand-driven dictionary construction is the subject of this section. The methods to be compared are DAPPER and Intermediate Signature Collection (ISC), since these are the two for which demand-driven diagnosis is feasible.

Start-up costs, information content, and extensibility of the two methods remain unchanged from section 8.4. When a dictionary is constructed gradually, both run-time and eventual total costs are important With multiple fault models, the dictionary is more a tool for recordkeeping than for performing diagnoses, so single fault models will be considered here.

8.5.1 Run-Time and Long Term Costs

The run-time costs are identical to those described in section 8.4.2.1 for both ISC and DAPPER, with the exception that repeated simulations are unnecessary, since the results may be obtained from the dictionary. Assuming that over time, all faults eventually occur will result in a kth fail dictionary eventually being constructed for DAPPER, at a cost equivalent to that reported in section 8.3.1.

For ISC, however, the long-term costs depend on the diagnosis strategy chosen. In a simple brute-force approach, eventually each fault could be simulated over each L pattern block, for a total effort equivalent to the cost of a signature by simulation dictionary As in section 8.3.1, this represents much more effort than required by DAPPER On the other hand, it offers the potential to locate some intermittent faults which are active only during some L pattern blocks, but not others. Since such faults must be active throughout the block to locate them with their signature, the benefits are probably limited

On the other hand, the number of faults to be simulated in a given block could be reduced if those known to fail in an earlier block are eliminated first. In this case, the total effort will depend on the order in which faults occur but could conceivably be compatible with that of k-Max DAPPER The relative positioning will depend on k and the distribution of detection probabilities within the circuit, with lower values of k and a tendency towards lower detection probabilities favouring DAPPER.

Chapter 9 Experimental Results

This chapter describes several experiments designed to analyze the performance of DAPPER. The circuits chosen are the 74LS181, a 4-bit arithmetic-logic unit (ALU) belonging to the TTL circuit family, and a set of "benchmark" circuits introduced by Brglez and Fujiwara at ISCAS-85 [Brg85]. These circuits have become the standard for analyzing the performance of testing algorithms and methods for combinational circuits. A description of the circuits is given in table 9.1:

Circuit	Inputs	Outputs	Gates
alu	14	7	58
C432	36	7	160
C499	41	32	202
C880	60	26	383
C1355	41	32	546
C1908	33	25	880
C2670	233	140	1193
C3540	50	22	1669
C5315	178	123	2307
C6288	32	32	2416
C7552	207	108	3512

Table 9.1 Circuits Used in Experiments

The number in the name of each ISCAS circuit refers to the number of lines it contains, while the "C" stands for *combinational* circuit. An additional circuit in [Brg85], C17, was not used in the experiments because of its small size.

The fault model used throughout this chapter is the single stuck-at fault model, because of its simplicity and its applicability to the tool sets available, and also to facilitate comparisons with other published results, most of which use the model.

9.1 Diagnosability with Detection Probability

The objective of this experiment is to estimate the performance of detection probability as a predictor (see chapter 5) While section 7.1.1 states that the accuracy of an error weight count as an estimate of detection probability increases with test length, at some point this coarse resolution will be unable to distinguish between some faults, even with increasing test length, because the faults are either equivalent or have identical detection probabilities.

In this experiment, the entropy function of [Man64], described in section 2.2.3, is used to determine diagnosability values of the weight count alone for the benchmark circuits. In this instance, the diagnosability function is given as:

$$R = -\frac{1}{\log h}\sum_{i=1}^{n} g(i)\log(g(i)) \tag{9.1}$$

where

$$\begin{aligned} n &= \text{test length} \\ h &= \text{total number of faults} \\ g(i) &= \frac{1}{h} \times \text{number of faults with count } i \end{aligned}$$

and the indeterminate $0 \log 0$ is defined to be 0.

This function has minimum value 0 when all faults have the same count, and maximum value 1 when each has a different count. As an example of the meaning of intermediate values, let h faults be grouped into $\frac{h}{k}$ sets of k elements each. Algebraic manipulation of (9.1) yields the following expression for R:

$$R = 1 - \frac{\log k}{\log h}$$

Values for various k and h are given in figure 9.1. It can be seen that inherent diagnosability of equivalently sized groupings decreases more slowly when there are more faults and hence more groups

9.1.1 Set-Up and Procedure

The inherent diagnosability for a test of length n is obtained by using the diagnosability function on the expected weights for each fault. In an ideal experiment, exact

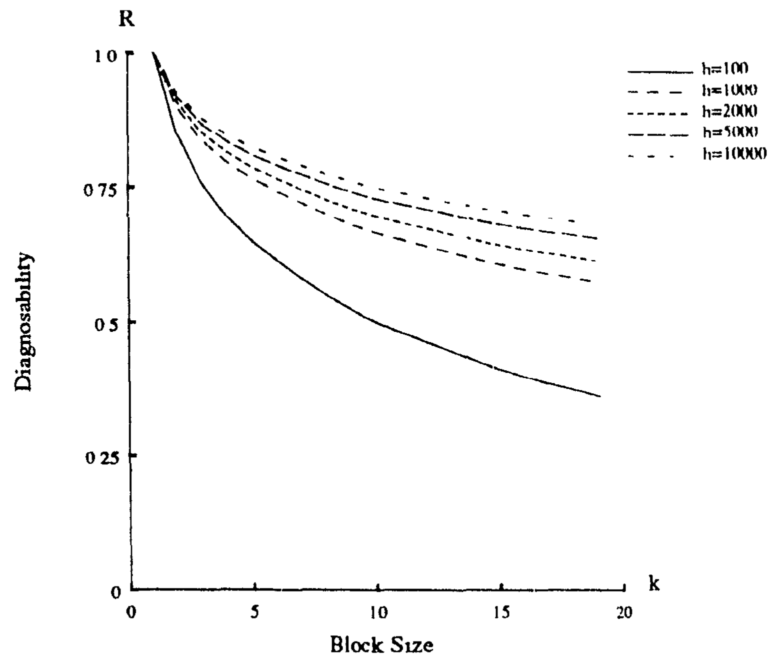


Figure 9.1 Diagnosability versus Size of Groupings

detection probability values would be available for each fault, and these could be used to calculate diagnosabilities for a variety of test lengths. Unfortunately, calculation of exact detection probability values is a #P-complete problem (see chapter 5), so, approximate values from simulation were used instead. In this modified experiment, expected counts were calculated for each fault for various test lengths, ranging in powers of 2 from 2^3 to 2^{20}, using both the independent and asymmetric error models. The diagnosability was then calculated from these.

Some error arises from these approximations. For example, faults with identical detection probabilities may have been estimated as close, but not equal (e.g. 0.505 and 0.495 for an actual value of 0.5). In this case, a test length of 1000 might be considered ample to distinguish the two faults, whereas in reality they could never be distinguished via detection probability alone. This is likely to be a more common effect in the independent error model than the asymmetric, since the latter's test-order dependencies make exact equivalence of detection probability highly unlikely for faults which affect individual outputs differently. Hence, the actual performance of detection probability as a predictor may be less than that expected, especially when the independent error model is used to estimate weight counts.

On the other hand, when detection probabilities are determined by statistical methods, so-called hard faults will be estimated to have detection probability 0. Long test

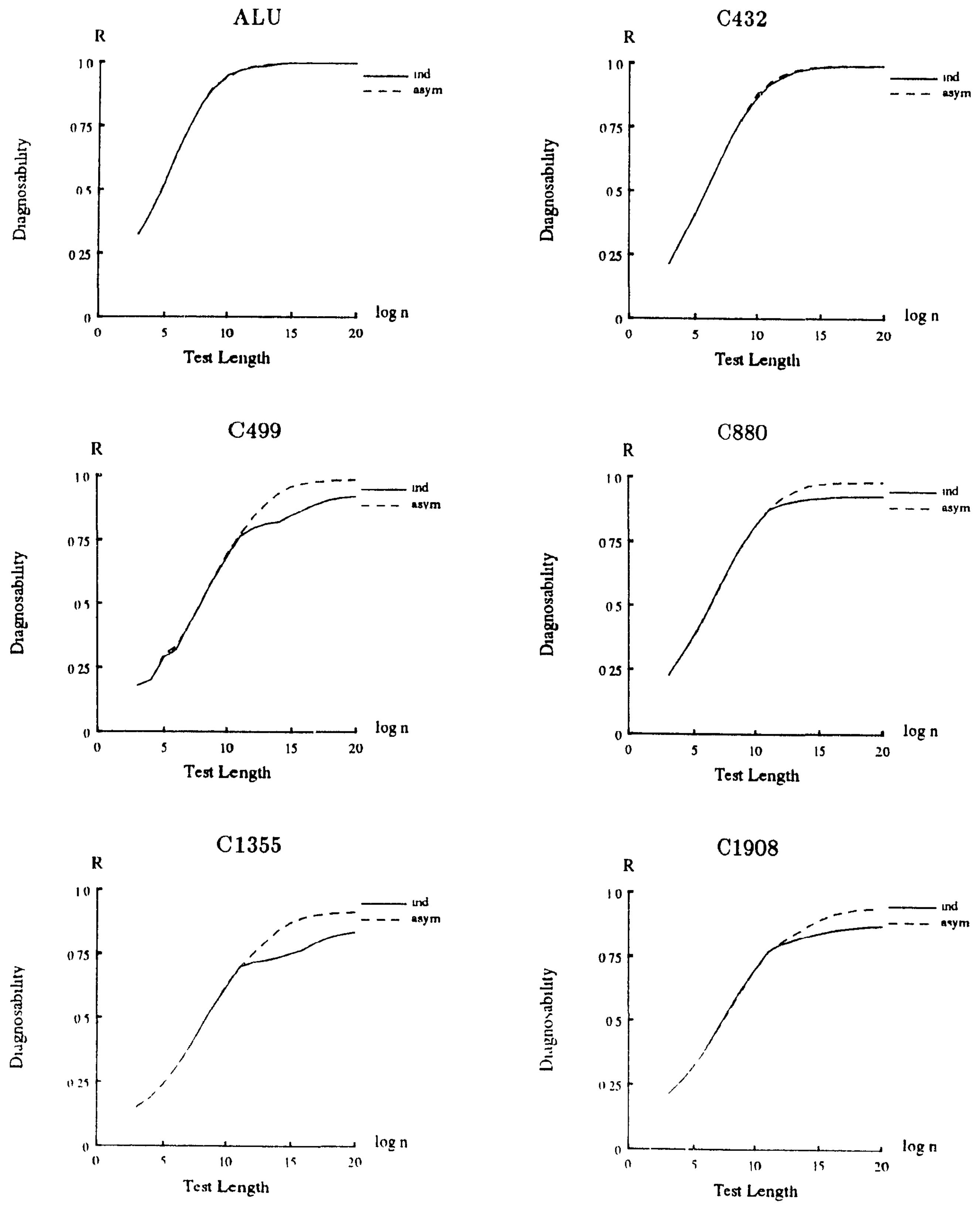
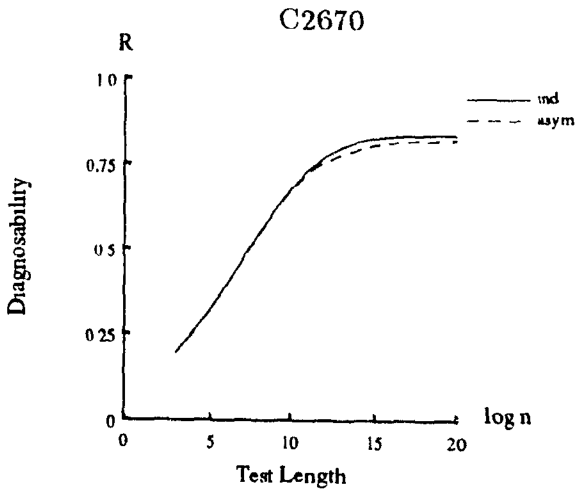
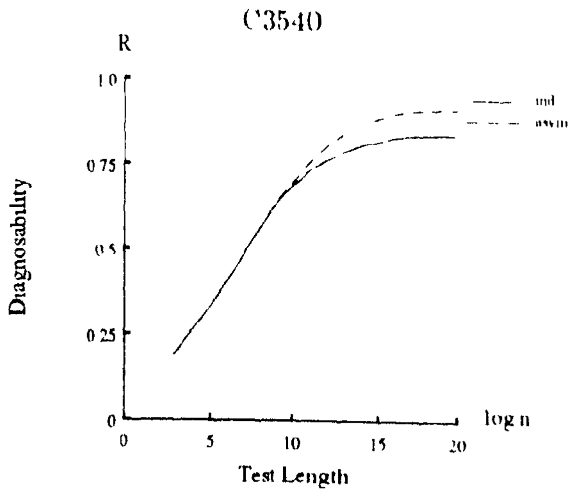
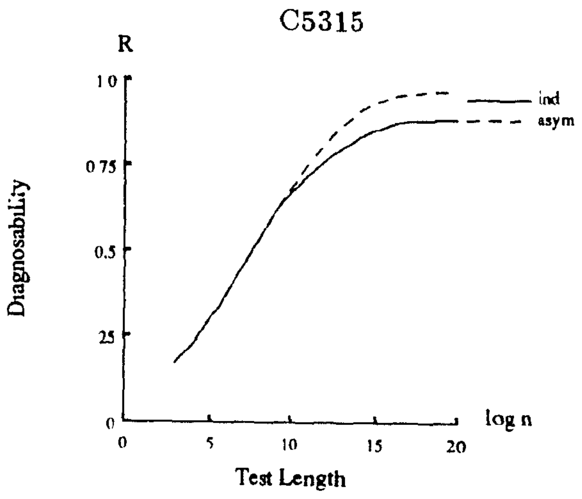
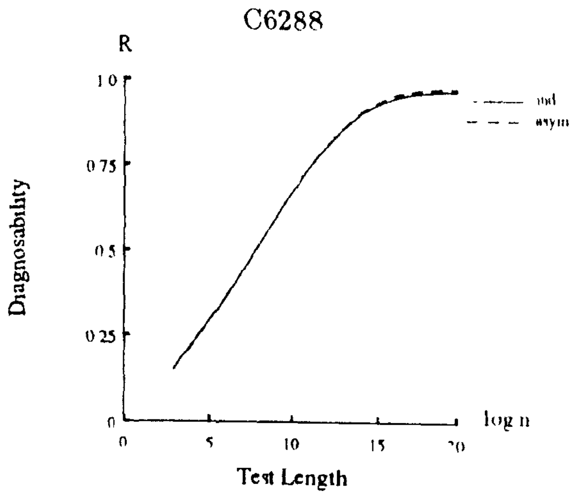


Figure 9.2 Diagnosability Functions, part 1









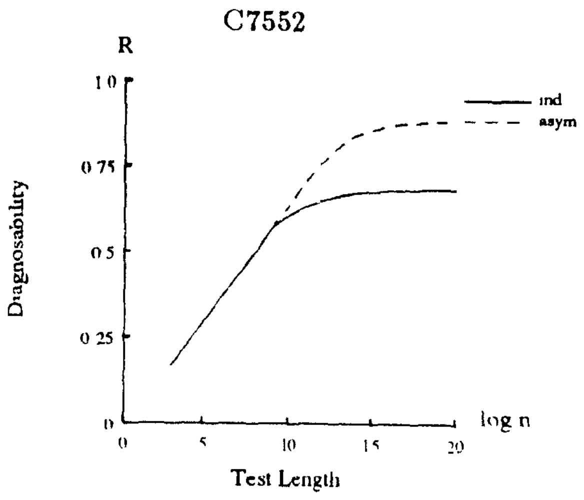


Figure 9.3 Diagnosability Functions, part 2

lengths may distinguish these faults, hence the inherent diagnosability for long test lengths may tend to be somewhat higher than suggested by this experiment.

The results also depend to some degree on the set of faults chosen. Inherent diagnosability will be lessened by an amount related to the number of redundant faults included, since none of these may be distinguished. Similarly, if equivalent faults are included, the diagnosability will decrease. In this experiment, equivalent faults which could be determined by inspection were not considered, while all redundant ones were in order to reflect the difficulty in identifying them.

Finally, the diagnosability function itself is somewhat misleading in its values Faults with detection probability 0.48 and 0.52 can be distinguished by an error count alone with 95% confidence only with test lengths greater than about 2000, while the diagnosability function will claim a distinction when n is greater than 25. Thus, diagnosability R is likely to be an overestimate of the true resolution capability of error weight counts.

9.1.2 Results

The results of the experiment are shown in figures 9.2 and 9.3. Several statements may be made about these results. First, the values rise rapidly from 0 (linearly when plotted on the logarithmic scale), then flatten out and approach some limit value. For many of the benchmark circuits, this limit is 1, while for others it can be as low as 0.7. This behaviour means that diagnosability increases with n for small test lengths, as faults with different detection probability are able to be distinguished. At some point, however, further distinction is impossible because of identical detection probability values.

It is interesting to note that the slope of the detectability rise is approximately constant among all the circuits. The rise also tends to level off between 2^9 and 2^{12}. It appears that this is due to the number of faults within the circuit, and that the levelling off commences when the majority of the faults achieve a unique predicted weight. As mentioned previously, fewer faults may achieve this when the independent error model is used, as can be seen for circuit C7552 in figure 9 3

A second feature of the results is that diagnosability tends to be higher when the asymmetric error model is used to predict results than when the independent error model is used These results are also consistent with expectations, since the asymmetric model allows a larger variety of probability values and hence the linear region should continue for a longer time The one exception to these results is circuit C2670, where the independent peak result is slightly higher. The cause of this behaviour is not immediately clear, although it may be due to the large proportion of undetected faults in the detection probability estimates.

9.1.3 Conclusions

It appears that diagnosability based on error weight counts alone is determined by circuit topology and distribution of detection probabilities. Function R is likely to be a loose upper bound of actual performance, so it is clear that detection probability alone will never be enough to completely characterize a fault. Nonetheless, fault resolution capability using error weight counts, and hence their utility as predictors, is likely to improve with test length, up to the point where those faults which have different detection probabilities have been separated.

9.2 Accuracy of Weight Prediction

The primary objective of these experiments is to determine how accurately the error weight counts used by DAPPER may be predicted for actual circuits, and hence the potential utility of the expected count permutation described in chapter 5. Secondary objectives include investigating the relationship between test length and prediction accuracy, and determining the effect of both partial matching and error in detection probability estimates on DAPPER predictions.

9.2.1 Set-Up

The circuits used for the experiments are again the 74LS181 ALU and the ISCAS benchmark circuits of [Brg85] (with the exception of C17). Input vectors were generated by autonomous primitive LFSRs, with connections as indicated in [Bar87] and initialized to random values, for all circuits except C1355 and C6288. Early results showed that the correlation in input patterns led to correlated output values for these circuits. DAPPER relies on successive output patterns being independent, and for these two circuits the LFSR-generated sequences could not provide this independence. To alleviate this problem, primitive cellular automata initialized to random values and with characteristic polynomials from [Bar87] were used instead for the two circuits. The configuration rules for the cellular automata were provided by M. Serra [Ser89], and developed using the technique reported in [Ser88].*

* It is interesting to note that C499, which is identical functionally to C1355, was not so adversely

The detection probability values were estimated from simulation on a set of purely random test vectors, in order to minimize possible effects of errors in their values While such Monte Carlo estimations cannot be used to produce meaningful absolute bounds on detection probability, they do provide confidence intervals. Finally, fast fault simulation tools available at McGill [Maa88] [Maa90] could be readily applied to estimating detection probabilities, while implementations of algorithms other than COP and that of [Kri86] were unavailable.[†]

9.2.2 Procedure

The experiments performed were as follows: Fixed test length simulations of DAPPER were performed on all the benchmark circuits. Simulations for various test lengths were performed up to an exhaustive test for the ALU, and to full fault coverage for one of the smaller benchmark circuits (lack of resources prevented this experiment from being performed on all the circuits), and finally the ALU was re-analyzed using detection probabilities estimated by COP as well as various amounts of random vector simulation In each case, the experimentally observed weight count was plotted against that predicted in chapter 7, and a line was fitted using least-squares regression (see for example [Won77]) The expected slope of the line is 1, and its expected intercept is 0. The experimental results are now reported in detail

9.2.3 Fixed Test Length

In these experiments, DAPPER was simulated on all the benchmark circuits for a fixed test length of 2048.[*] This length, while not sufficient to detect all faults in many of the circuits, was the maximum possible for full simulation of all circuits given computing resource constraints. The number of undetected faults using the test set for each circuit is shown in table 9 2, together with the number of redundant faults (from [Sch88]) The difference between these two figures gives the number of detectable faults missed by the test set. The last column of the table gives the irredundant fault coverage of the test set

affected by the correlated outputs C499 is an XOR gate implementation of a parity circuit, while C1355 is a NAND implementation Apparently, the faults within C1355 which cause the problems are those internal to the exclusive-or structure and hence not covered in the fault set of C499

[†] See section 2 5 3 for additional information on testability measures The effects on count predictions of COP and simulation experiments for estimating detection probability are discussed later in section 9 2 5

[*] Some of the results in this section were originally reported in [Ait89a]

Circuit	Faults	Undetected	Redundant	Missed	Irred. Cov.
alu	237	0	0	0	100.0
C432	524	4	4	0	100.0
C499	758	8	8	0	100.0
C880	942	9	0	9	99.0
C1355	1574	8	8	0	100.0
C1908	1879	29	9	20	98.9
C2670	2595	431	117	314	87.9
C3540	3428	156	137	19	99.4
C5315	5350	60	59	1	99.9
C6288	7744	34	34	0	100.0
C7552	7548	494	131	363	95.2

Table 9.2 Undetected Faults After 2048 Vectors

The test sets for the ALU, C432, C499, C1355, and C6288 detected all the irredundant modelled faults, while that for C2670 detected only 88% of these. The existence of missed faults will not adversely diagnosis of the detected faults, since the missed faults have very low detection probabilities and are consequently expected to be missed by a test of length 2048 Naturally, a fault which is never detected cannot be located by any method.

Each experiment proceeded as follows: The circuit was simulated fault-free to produce a quotient sequence and fault-free LFSR signature. This quotient sequence was then matched with the simulated output of the MISR for each fault, and the three signatures (weight count, first-fail, and LFSR) were obtained

The weight count was then compárared with that predicted by both equation (7 1 1) for the independent error model and equation (7 1 8) for the asymmetric error model The comparison proceeded by using a least-square line to fit a plot of the observed versus predicted value for the two models. For the expected count permutation to be useful as a predictor, these predictions must be close to experimentally observed values

9.2.3.1 Standard DAPPER

In the first part of the experiment, 100% matching of the fault-free MINSR sequence was used to obtain the weight counts. The values for the slope and intercept of the fitted line, together with the sample standard deviation for its data points,* are given in table 9.3 for both error models.

Circuit	Model	Slope	Intercept	S-DEV
alu	ind	0.982	7.5	42.9
	asy	0.988	6.3	40.0
C432	ind	0.981	3.1	40.8
	asy	0.988	3.2	39.3
C499	ind	0.987	4.9	23.5
	asy	0.995	1.6	14.4
C880	ind	0.987	-0.5	27.9
	asy	0.999	1.2	22.6
C1355	ind	0.988	3.9	18.0
	asy	0.997	3.6	14.1
C1908	ind	0.995	4.5	25.0
	asy	1.001	4.5	23.8
C2670	ind	0.985	-0.4	22.9
	asy	1.000	-0.5	20.5
C3540	ind	0.986	4.1	42.1
	asy	0.991	1.1	41.5
C5315	ind	0.966	1.5	36.1
	asy	0.996	0.4	32.2
C6288	ind	0.992	1.3	37.4
	asy	0.999	1.0	37.0
C7552	ind	0.961	4.3	34.7
	asy	0.993	4.0	33.3

Table 9.3 Accuracy of DAPPER Weight Predictions, 100% Matching

As an example of the results of table 9.3, consider circuit C880. Observed counts y_i

* The sample standard deviation is calculated as follows

$$\text{S-DEV} = \sqrt{\frac{1}{h-1}\sum_{i=1}^{h}(ax_i + b - y_i)^2}$$

where a and b are respectively the slope and intercept of the fitted line h is the number of faults, x_i is the predicted count for fault i and y_i is the observed count are related to counts predicted by the independent error model x_i by the equation:

$$y_i = 0.987x_i - 0.5$$

with a sample standard deviation of 27.9. When the asymmetric error model is used to predict counts, the equation changes to:

$$y_i = 0.999x_i + 1.2$$

and the sample standard deviation decreases to 22.6

The results of table 9.3 show that DAPPER weight predictions are consistent with both the independent and asymmetric models. It also appears that the asymmetric is a slightly better predictor, since its slope is in all cases closer to 1 than that of the independent error model. The intercept values are not always closer to 0, but the standard deviation of the observed points is always lower. One reason for this behaviour is that the asymmetric model can predict the weight of faults on primary outputs perfectly, since their output stream is deterministic under the model

9.2.3.2 Effect of Partial Matching

The experiments of the previous section were repeated with partial matching applied to the MINSR stream. The results are shown in table 9.4

Table 9.4 gives the slope, intercept, and standard deviation of the fitted line for both the independent and asymmetric error models for a variety of partial matchings with the benchmark circuits. The fraction of 1s and 0s in the fault-free stream matched are given separately (although the first F bits were fixed at 100% matching, as suggested in chapter 7 to make the first-fail value useful, with F being the maximum of 32 and the number of circuit outputs). In addition, the fault free weight of the output stream is given, along with the number of faults whose weight counts aliased. In no case did all three signatures alias

Note that in general the aliasing increases as the count increases, as expected given the results of appendix A. In addition, the narrowing of the range of potential weight counts results in lower standard deviations as matching decreases, although cutting the range in half with 75% matching does not always reduce the standard deviation by a corresponding amount. Finally, the asymmetric error model again outperforms the independent in terms of standard deviation. It can be seen that partial matching does not greatly reduce the ability to predict values for the error weight count

Circuit	1-match	0-match	Model	Slope	Intercept	S-DEV	w	aliased
alu	0.9	1	ind	0.978	11.5	39.4	95	0
			asy	[illegible]	8 0	36 1		
	0.75	1	ind	[illegible]	15.3	32.7	251	0
			[illegible]	[illegible]	8.6	30 9		
	0 75	0 75	[illegible]	0 986	10 1	21 9	537	0
			asy	0.992	7 6	20 8		
C432	0.9	[illegible]	ind	0 984	5.4	36.3	111	0
			asy	0 991	1.4	35 0		
	[illegible]	0.75	ind	0.985	7.8	22.0	520	0
			asy	0.994	1 5	20.1		
[illegible]	0 9	1	ind	0 986	6.6	22 0	102	0
			asy	0.995	5.5	13.4		
C880	0.9	1	ind	0 986	0 9	25 9	102	0
			asy	0 999	1 2	21 2		
C1355	0.9	1	ind	0.988	5.0	17.8	96	0
			asy	0 996	1 0	13.6		
C1908	0.9	1	ind	0.996	5.1	23.4	98	3
			asy	1.001	4.9	22.0		
C2670	0 9	1	ind	0.985	1.1	21.8	92	0
			asy	1.000	-0.7	19.8		
C3540	0 9	1	ind	0 986	5 2	39.2	98	2
			asy	0.990	5 0	38 5		
	0.75	0.75	ind	0.985	9.0	23 4	537	15
			asy	0.991	6.2	23 3		
C5315	0.9	1	ind	0.967	1.9	33 7	98	0
			asy	0 996	0 8	30 4		
	0 75	0 75	ind	0.966	18 5	22 6	537	17
			asy	0 997	1 5	20 6		
C6288	0 9	1	ind	0 992	1 3	34 9	100	0
			asy	1 000	0 4	34 5		
C7552	0 9	1	ind	0 963	7 4	3[illegible] 7	102	1
			asy	0 993	1 2	30 4		
	0 75	0.75	ind	0 955	26.0	21 2	537	29
			asy	0 990	6.9	20 3		

Table 9.4 Accuracy of DAPPER Weight Predictions Partial Matching

9.2.3.3 Scan DAPPER

This experiment investigates the effect of the parallel to serial (P/S) compaction performed by the MINSR on the performance of the expected weight count permutation, as well as the potential effects of cancellation aliasing on fault detection. A control experiment was set up using Scan DAPPER with serial streaming (see section 7.2) instead of standard DAPPER's MINSR. The test was conducted with n_1 set to n in order to minimize any error in the results. The weight prediction experiment was repeated using serial streaming and the results are shown in table 9.5.

Circuit	Model	Slope	Intercept	S-DEV	Max(w)	Extra
alu	ind	1.002	4.7	59.3	1055	0
	asy	1.000	4.0	50.5		
C432	ind	0.976	6.1	54.0	6197	0
	asy	0.982	7.3	47.6		
C499	ind	0.988	18.0	30.4	1090	0
	asy	0.989	18.7	21.9		
C880	ind	0.998	1.4	29.1	3267	0
	asy	1.008	0.9	24.9		
C1355	ind	0.990	16.0	21.5	4077	0
	asy	0.991	15.7	17.8		
C1908	ind	1.005	9.0	33.4	5775	0
	asy	1.009	9.0	31.8		
C2670	ind	1.000	1.6	38.2	10721	1
	asy	0.999	3.1	35.2		
C3540	ind	0.990	11.7	85.8	7144	0
	asy	0.994	10.7	84.0		
C5315	ind	1.002	1.2	70.1	18144	0
	asy	1.000	1.1	62.8		
C6288	ind	1.006	8.7	97.7	9512	0
	asy	1.006	9.3	97.3		
C7552	ind	1.002	3.9	63.5	19270	[illegible]
	asy	1.003	3.6	62.0		

Table 9.5 Accuracy of DAPPER Weight Predictions Serial Streaming

Table 9.5 again gives slope, intercept, and standard deviation for the two error models. It also includes the maximum weight counter value observed for any fault in the circuit. The final column is the number of "extra" faults; that is, those found by the serial streaming technique yet missed by the MINSR. Only 2 circuits have such faults. In each case, the fault was detected by a pattern near the end of the test sequence and had not yet been shifted out of the MINSR. There was no example of cancellation aliasing in any of the circuits, lending credence to the claim in section 7.4 that it is a rare occurrence

The results of the experiment indicate that the accuracy of the predictions using serial streaming is typically somewhat higher than those with the MINSR, as indicated by a comparison of the slope and intercept values between tables 9 3 and 9 5. The increased range results in increased standard deviation. Any improvement in accuracy is likely due to the fact that calculating expected weight with serial streaming involves only adding estimated probabilities, while with a MINSR these must also be multiplied, as in algorithms 6.2 and 6.3, which tends to compound numerical error

The asymmetric error model again generally outperforms the independent, but the difference is very slight for circuits C2670, C3540, C6288, and C7552. In conclusion, it seems that using a MINSR results in only a slight reduction in the performance of the expected count permutation, and that cancellation aliasing in the MINSR is not a serious problem in DAPPER.

9.2.4 Variable Test Length

These experiments were performed to illustrate the effect of test length on the accuracy of DAPPER's performance. The 74LS181 ALU was simulated for test lengths up to 2^{14}, which is an exhaustive test, while benchmark circuit C880 was simulated on lengths up to 2^{15} (full fault coverage was obtained at length 2^{14}) Test vectors were generated by autonomous LFSRs, for test lengths in powers of two, starting from 64 for the ALU and 128 for C880.

The results for the ALU are plotted in figures 9 4 through 9 6 The slope approaches 1 with increasing n for the independent error model, but remains roughly constant at 1 for the asymmetric model This may be due to the fact that the asymmetric error model accounts for discrepancies in probability in the first m vectors, where m is the number of circuit outputs (these differences are discussed in section 6.1 2). The affect of these values decreases with test length and is reflected in the slope figures

The intercept values (normalized by test length) are shown in figure 9.5 They show a somewhat puzzling behaviour: Dropping until n reaches 512 then increasing The

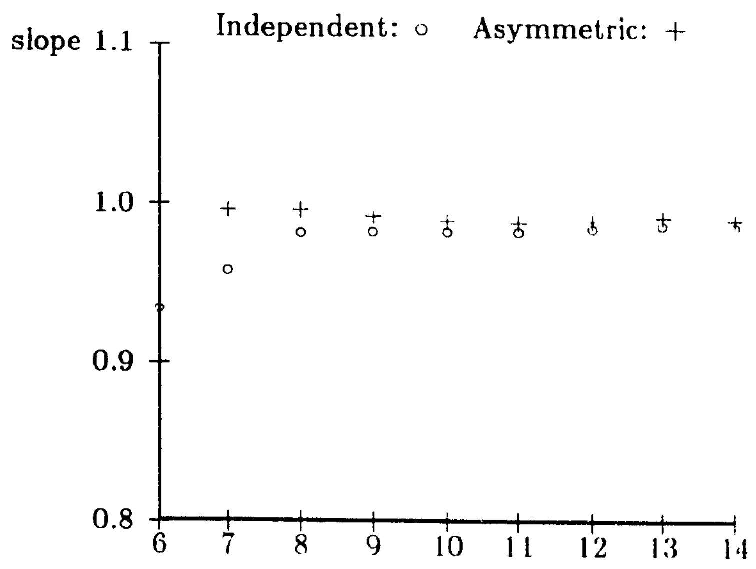


Figure 9.4 Slope versus $\log_2$ Test Length, 74LS181 ALU

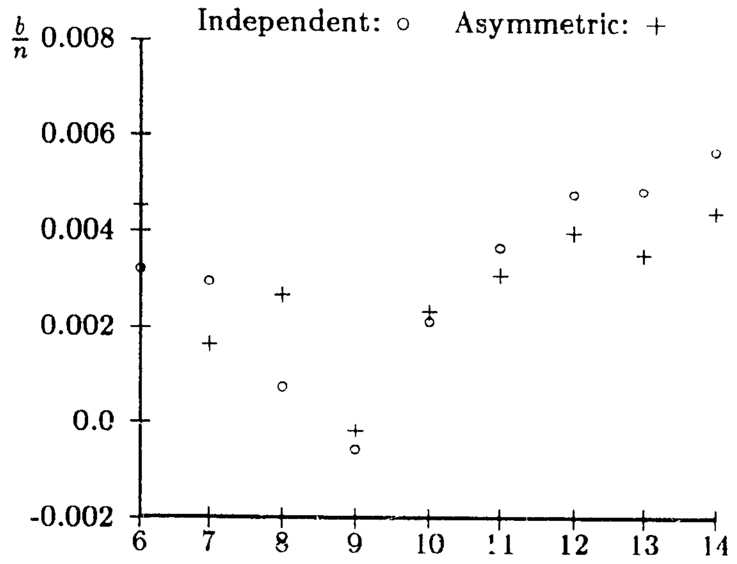


Figure 9.5 Normalized Intercept versus $\log_2$ Test Length, 74LS181 ALU

reasons for this behaviour are unclear

The normalized standard deviation is shown in figure 9 6. It decreases rapidly at first, then levels off. Recall that it is expected that this value will decrease as the square root of test length, so with each doubling of test length, the normalized standard deviation should drop by about 30%. This decrease, however, requires precisely accurate detection probability values, not estimates The error inherent in the detection probability estimates prevents the standard deviation from dropping below a certain threshold and produces the slowing affect observed in figure 9 6.

Circuit C880 exhibits similar behaviour to the ALU, as evidenced by figures 9 7

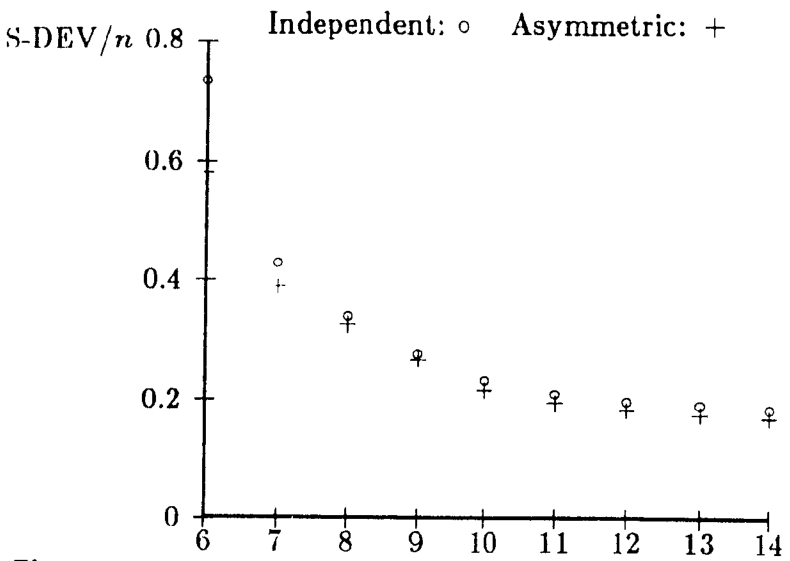


Figure 9.6 Normalized Standard Deviation versus $\log_2$ Test Length, 74LS181 ALU

through 9.9. The slope again approaches 1, the normalized standard deviation again drops rapidly at first then slows as the inherent error in detection probability estimates begins to significantly affect results. The intercept values of figure 9.8 also drop then rise, as in figure 9.5, but the change is not as pronounced. Again, the reasons for this shape are unclear.

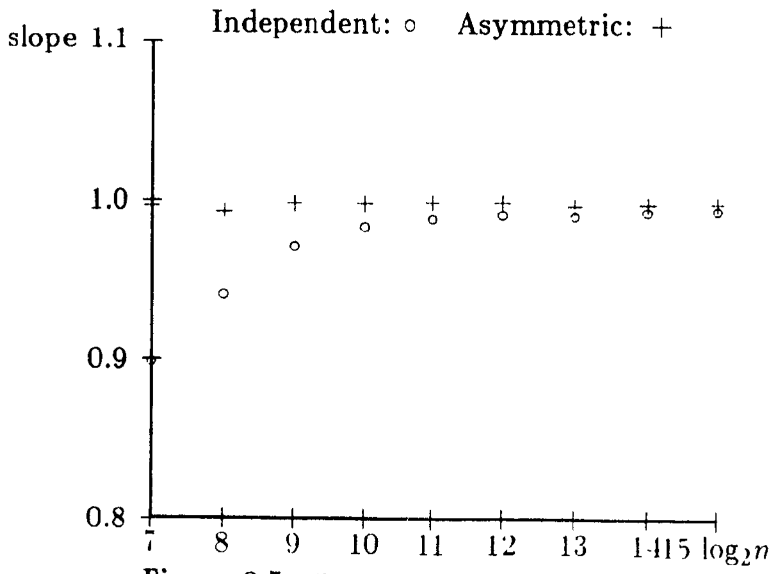


Figure 9.7 Slope versus $\log_2$ Test Length C880

Thus, the expected count permutation's performance as a predictor tends to improve with test length, but this improvement eventually slows as inaccuracies in detection probability become more significant. These results can be capitalized upon by k-Max DAPPER, since k can be set in such a way as to maximize the performance gains while

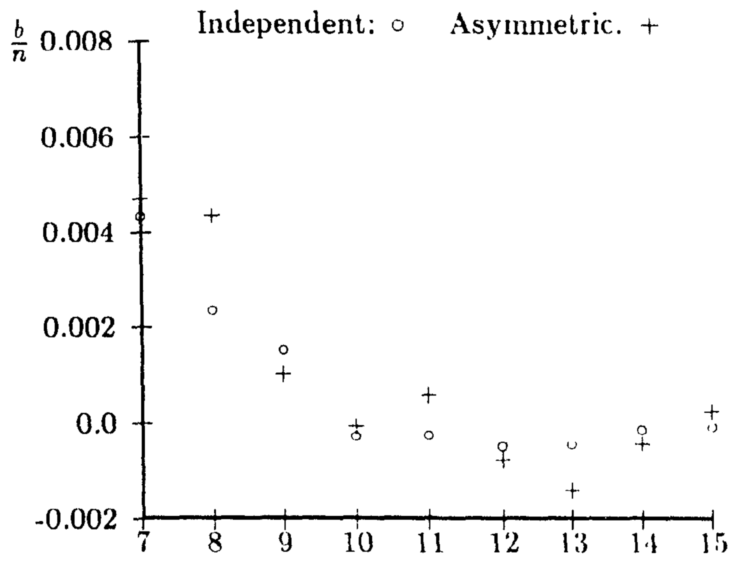


Figure 9.8 Normalized Intercept versus $\log_2$ Test Length, C880

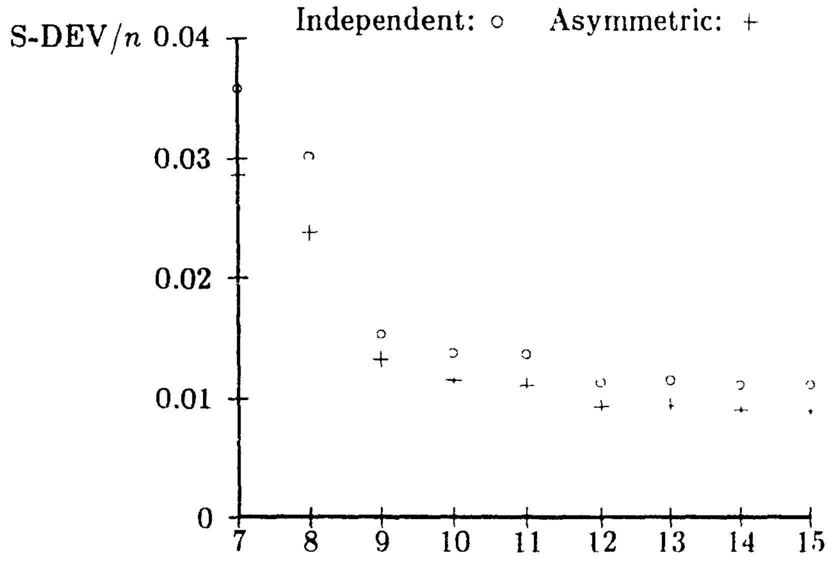


Figure 9.9 Normalized Standard Deviation versus $\log_2$ Test Length, C880

minimizing the test length for each fault.

9.2.5 Effect of Detection Probability Estimation

In these experiments, detection probability estimates were obtained in two ways. First, using COP, and second by estimation from simulation on test sets of random length. The circuit used was the 74LS181 ALU, and the estimation sets were of lengths ranging from 64 to 16384 in powers of two. For example, if a fault was detected 55 times on one output in a test length of 256, its detection probability for the independent error model was estimated to be 55/256 or 0.215. Similarly, if 117 of the 256 output values were 1 in the fault free case, and errors occurred on 37 of them, p_D was estimated to be 37/117 or 0.316, while $p_{\overline{D}}$ would be 18/139 or 0.129. The results of fitting the predictions to observed values for the test of length 2048 are shown in table 9.6

Sim Len.	Model	Slope	Intercept	S-DEV
COP	ind	0.881	35.547	148.5
	asy	0.948	-14.012	133.7
64	ind	0.935	26.438	105.2
	asy	0.938	29.923	95.6
128	ind	0.982	0.716	72.9
	asy	0 980	6.578	63.1
256	ind	0.988	-0.929	60.0
	asy	0 985	4.670	51.3
512	ind	0.991	0 944	49.3
	asy	0.988	6 272	43.0
1024	ind	0.981	9.584	45.6
	asy	0.986	9.502	41.4
2048	ind	0 982	7 447	42 9
	asy	0.988	6.310	40.0
4096	ind	0.985	4.749	37.4
	asy	0 989	5 009	36 0
8192	ind	0.986	2.381	35.9
	asy	0.990	2.870	34.4
16384	ind	0 988	0.701	35.8
	asy	0.991	1 898	34.3

Table 9.6 Accuracy of DAPPER Weight Predictions, Various Probability Estimates

Table 9.6 shows that the statistical methods are better weight predictors than COP, in terms of observed standard deviation, even when only 64 vectors are simulated per fault Nonetheless, COP's linear processing time may offset its inaccuracy, especially when fast fault simulation tools are unavailable. In addition, large amounts of simulation are not required before an acceptable accuracy in detection probability estimates is acheived — certainly the performance improvement between 512 and 16384 does not appear to justify a 32-fold increase in simulation.

9.2.6 Results

It has been shown that the expected weight count permutation is able to accurately model fault behaviour. Prediction is improved with increasing test length and more accurate estimation of detection probability. In general, the asymmetric error model is a better at predicting observed error weight counts than the independent error model, although the difference is occassionally minimal.

9.3 Effort Required for Fault Resolution

The objective of these experiments is to examine both the amount of post-test fault simulation required by DAPPER for individual fault resolution with demand-driven fault location techniques and the potential savings over signature by simulation methods if a dictionary is constructed in advance An additional objective is to determine the value of the three signatures in avoiding misdiagnosis. As in section 9 2, analysis is made when test length is fixed, when it is varied, and for various detection probability estimates.

9.3.1 Set-Up

The set-up for these experiments is identical to that of the weight prediction experiments in section 9 2.

9.3.2 Procedure

Individual fault location to fault equivalence classes was performed in these experiments. The trials used are identical to those of section 9 2

Recall that demand-driven fault location with data compaction proceeds by post-test fault simulation. First, a diagnosability profile (a list of faults sorted by their predicted weights) is computed. Next, the observed weight from the test is noted, and the fault whose predicted weight lies closest to the observed value is simulated to its first failing pattern, or the observed first failing pattern, whichever is less If these match, simulation continues for the remainder of the test length (in standard DAPPER or until the weight count reaches k in k-Max DAPPER), to obtain a weight count and LFSR signature If these two signatures also match, the fault has been located Otherwise, the fault with the next closest predicted weight is tried, and so on.*

* No termination condition was employed in these experiments, although in practice simulation could

In order to identify misdiagnosis, these experiments were modified as follows: A fault was simulated to produce an observed weight. DAPPER was allowed to resolve the fault using the method described above, but any occassion where all signatures matched before the simulated fault was reached was noted. Typically, such a misdiagnosis is caused by identical output values for two faults. Of course, this situation precludes any method from resolving two faults.

Comparisons between the simulation effort required were made on the basis of assumption 2 1; that is, that simulation effort can be measured on the basis of number of fault-patterns simulated.

9.3.3 Fixed Test Length

The test sets used for the fixed length experiments were identical to those in section 9.2.3. Table 9.2 describes their characteristics. Diagnosability profiles were constructed for these trials, and fault location performed for each of the faults. The order of subsections in this section matches that of section 9.2.3.

9.3.3.1 Standard DAPPER

The average amount of simulation required to resolve a single fault for each of the benchmark circuits for standard DAPPER is reported in table 9.7.

Table 9 7 gives several results for each circuit: the effort required to construct a first-fail dictionary in advance as a percentage of the cost of a full (2048 pattern) dictionary for each fault, the average percentage of faults whose first-fail values are investigated for each location (h_1), the average number of faults which must be simulated completely to locate each fault (h_2), and finally the effort that this simulation represents as a percentage of the cost of a full dictionary construction, both with and without the prior construction of a first-fail dictionary. These values are determined using both the independent and asymmetric error models to predict detection probability.

Thus, for circuit C5315 when the independent error model was used to predict weights, locating an individual fault required, on average, simulation up to the first failing pattern for 8 3% of the 5290 faults On average 1 08 faults remained after this stage This effort, plus continued simulation to distinguish the faults took 0 27% of the simulation effort required to construct a full fault dictionary (using the fault-pattern assumption of section 5 1.1) Had a first-fail dictionary been constructed in

stop once the predicted weights being investigated were sufficiently far from the observed value. At this point, an unmodelled fault would be declared to have occurred

Circuit	Faults	1st Fail Set-up (%)	Model	h_1 (%)	h_2	Effort (vs. Dictionary Constr.)	
						w/o 1st fail (%)	with 1st fail (%)
alu	237	0.84	ind	5.9	1.00	0.71	0.42
			asy	5.1	1.00	0.66	0.42
C432	520	3.16	ind	7.6	1.02	0.44	0.19
			asy	7.4	1.02	0.42	0.19
C499	750	4.99	ind	20.1	1.13	1.12	0.15
			asy	14.2	1.14	1.00	0.15
C880	933	5.42	ind	4.8	1.07	0.28	0.11
			asy	4.4	1.08	0.27	0.11
C1355	1566	9.12	ind	15.5	1.24	0.76	0.08
			asy	13.4	1.25	0.74	0.08
C1908	1850	12.88	ind	6.1	1.11	0.17	0.06
			asy	5.5	1.10	0.45	0.06
C2670	2164	19.19	ind	3.5	1.03	0.13	0.05
			asy	2.8	1.37	0.12	0.06
C3540	3272	8.45	ind	6.8	2.16	0.20	0.07
			asy	6.7	2.16	0.20	0.07
C5315	5290	4.76	ind	8.3	1.08	0.29	0.02
			asy	7.7	1.08	0.27	0.02
C6288	7710	1.50	ind	10.5	1.14	0.50	0.015
			asy	10.0	1.13	0.47	0.015
C7552	7044	10.81	ind	5.4	1.23	0.23	0.017
			asy	5.0	1.23	0.21	0.017

Table 9.7 Fault Resolution, Standard DAPPER

advance (which would require an effort of 4.76% of a full dictionary construction), the per location simulation effort would drop to 0.02% of a full dictionary construction. When the asymmetric model was used instead, h_1 dropped slightly to 7.7% of the faults, and h_2 remained at 1.08, resulting in a slight drop in overall effort to 0.27% of full dictionary construction. The cost when the first failing pattern dictionary was already in place remained constant at 0.02%.

It can be seen from table 9.7 that standard DAPPER is able to resolve a given fault with a small amount of effort. The size of fault classes affects this value, as evidenced by circuit C499, where many faults have detection probability of 0.5, and hence cannot be resolved effectively using DAPPER's weight count. It is also evident that the asymmetric error model is usually able to reduce the simulation effort slightly over that required by the independent error model. The costs when a first-fail dictionary is constructed in advance are reduced to such an extent that the look-up time for the dictionary may become significant. In addition, when only one fault is to be simulated, simulator overhead may become significant for some simulation tools. Thus, the savings indicated should be taken as an upper bound, rather than as an absolute guarantee.

Table 9.7 shows the importance of the first-fail counter in reducing the amount of simulation performed. The coarse resolution provided by the weight counter is important for reducing the number of faults which need to be investigated. Finally, the LFSR signature is required for individual fault location.

The next part of the experiment analyzes the effect of the three signatures on diagnostic resolution. In other words, how frequently is a misdiagnosis made if some of the signatures are not included. Two output sequences were assumed to be identical if their LFSR signatures, weight counts, and both first and second failing patterns matched, since storage of the entire output set for each fault would be infeasible. The following information was collected for each circuit:

NC: The total number of comparisons made for all faults.

NEQ: The total number of equivalent output pairs found during fault location.

NL: The total number of equivalent LFSR signature pairs found outside of NEQ.

NWL: The total number of output pairs with both equivalent LFSR signatures and weight counts found outside of NEQ.

NWF: The total number of output pairs with both equivalent weight count and first failing pattern found outside of NEQ.

NWLF: The total number of output pairs with all three signatures equivalent found outside of NEQ.

The values for these quantities are shown in table 9.8, and the values normalized by the number of faults in the circuit are given in table 9.9. The latter table gives the average number of such equivalencies found per fault. Only the independent error model is shown in the tables. The results for the asymmetric model are similar.

Several results are apparent from tables 9.8 and 9.9. First, the DAPPER method gives better diagnostic resolution than a 16-bit LFSR signature alone, as predicted in section 8.1, since NL is non-zero for 10 of the 11 circuits. Second, an LFSR signature is required, since weight count and first-fail together do not provide adequate diagnostic capability. The first-fail counter improves diagnostic resolution in only 3 of the circuits.

Circuit	NC	NEQ	NL	NWL	NWF	NWLF
alu	3.3e3	0	0	0	0	0
C432	2.1e4	4	5	0	8	0
C499	1.2e5	0	6	0	97	0
C880	4.2e4	15	1	0	50	0
C1355	3.8e5	254	26	3	127	0
C1908	2.1e5	67	9	0	130	0
C2670	2.4e5	59	10	0	5	0
C3540	8.0e5	3145	38	0	314	0
C5315	2.4e6	184	138	1	233	0
C6288	6.3e6	346	383	2	763	0
C7552	3 1e6	614	162	0	1032	0

Table 9.8 Total Equivalencies Found During Fault Location

Circuit	AC	AEQ	AL	AWL	AWF	AWLF
alu	14	0 000	0 000	0 000	0 000	0 000
C432	40	0.008	0 010	0 000	0 015	0 000
C499	152	0 000	0.008	0 000	0 129	0 000
C880	15	0.016	0.001	0.000	0.054	0.000
C1355	244	0.162	0 017	0 002	0 081	0 000
C1908	114	0 036	0 005	0 000	0 070	0 000
C2670	91	0 027	0 005	0 000	0 002	0 000
C3540	233	1.053	0.012	0.000	0 105	0.000
C5315	444	0 035	0 026	1 9e-4	0 044	0.000
C6288	810	0.045	0 050	2.5e-4	0 099	0.000
C7552	408	0 087	0 023	0 000	0 147	0 000

Table 9.9 Normalized Equivalencies Found During Fault Location

but its usefulness in reducing simulation has already been documented In addition in each case where all three signatures were equivalent between a pair of faults, the second failing pattern was also equivalent, lending support to the assumption that such sequences are identical for those cases considered Finally, only circuits C3540 and C1355 appear to have a proportionally large number of faults with equivalent behaviour over the test sets,* so misdiagnosis from equivalent signatures does not appear to be a serious problem for the circuits observed.

9.3.3.2 Effect of Partial Matching

The experiments reported in table 9.7 were repeated with partial matching using the examples discussed in section 9 2.3.2. The results are shown in table 9.10.

Table 9 10 does not list either the effort required to generate a first-fail dictionary or the savings which could be obtained by using it. Matching was fixed at 100% for the first F vectors, where F was set to the maximum of 32 and the number of circuit outputs In most cases, this value appears to be a reasonable compromise, in that the effort observed is not much worse than that seen with perfect matching in table 9.7. There are two notable exceptions to this, however: circuits C499 and C1355 In both these cases, partial matching eliminated the usefulness of the first fail signature, leaving the error count as the only hierarchical reduction, so that the total effort was not much less than h_1.

For the remainder of the circuits, it can be seen that as the matching level decreases, the total effort increases (with the notable exception of the third trial for the alu). In addition, the asymmetric model tends to be a slight improvement over the independent.

9.3.3.3 Scan DAPPER

These experiments are to determine potential savings in simulation when Scan DAPPER with serial streaming rather than P/S compaction, is used The experimental setup is identical to that of section 9 2.3 3, and again n_1 is set equal to n in order to analyze the maximum possible improvement. The potential failing output set was not used, instead the first failing output was determined from the first failing pattern Faults which were not physically connected to the first failing output were not simulated The results are shown in table 9 11

The results when a first failing pattern dictionary was assembled in advance are not included in table 9 11, since they are virtually identical to those in table 9.7 Table 9.11 includes an additional field over table 9 7, "% Change", which gives the percentage improvement of serial streaming over the P/S compaction results of section 9 3 3.1, when no first-fail dictionary was compiled The improvement ranges from nonexistent in the case of C1355 where removing the MINSR increased the average amount of simulation

* Recall that trivially equivalent faults such as AND gate inputs stuck at 0, were previously eliminated from the test sets

Circuit	0-match	1-match	Model	h_1 (%)	h_2	Effort (%)
alu	0.9	1	ind	6.0	1.00	0.73
			asy	5.0	1.00	0.68
	0.75	1	ind	6.1	1.00	0.73
			asy	5.4	1.00	0.71
	0.75	0.75	ind	5.9	1.00	0.71
			asy	5.2	1.00	0.69
C432	0.9	1	ind	7.5	1.03	1.03
			asy	7.2	1.02	0.99
	0.75	0.75	ind	9.1	1.03	1.25
			asy	8.5	1.03	1.19
C499	0.9	1	ind	20.4	1.81	8.99
			asy	14.5	1.94	8.75
C880	0.9	1	ind	4.8	1.08	1.20
			asy	4.5	1.09	1.17
C1355	0.9	1	ind	22.4	6.48	12.58
			asy	22.6	6.50	12.54
C1908	0.9	1	ind	6.4	1.84	1.92
			asy	5.6	1.82	1.89
C2670	0.9	1	ind	3.7	1.04	0.16
			asy	3.0	1.04	0.14
C3540	0.9	1	ind	6.9	2.18	1.15
			asy	6.8	2.19	1.15
	0.75	0.75	ind	7.7	2.35	1.24
			asy	7.7	2.35	1.25
C5315	0.9	1	ind	8.5	1.10	0.51
			asy	8.0	1.09	0.47
	0.75	0.75	ind	10.1	1.15	0.59
			asy	9.5	1.15	0.56
C6288	0.9	1	ind	10.6	1.16	0.58
			asy	10.1	1.15	0.54
C7552	0.9	1	ind	5.4	1.17	0.49
			asy	5.0	1.17	0.47
	0.75	0.75	ind	6.7	1.28	0.60
			asy	6.2	1.28	0.58

Table 9.10 Fault Resolution, Standard DAPPER, Partial Matching

Circuit	Model	h_1 (%)	h_2	Effort (%)	% Change
alu	ind	1.6	1.00	0.50	79.4
	asy	4.1	1.00	0.48	72.7
432	ind	6.7	1.01	0.28	63.6
	asy	6.4	1.01	0.27	64.3
499	ind	19.5	1.12	0.83	74.1
	asy	16.6	1.12	0.83	83.0
880	ind	4.6	1.05	0.11	39.3
	asy	4.1	1.06	0.11	40.7
1355	ind	17.0	1.45	0.83	109.2
	asy	15.8	1.43	0.82	110.8
1908	ind	5.6	1.08	0.11	23.4
	asy	5.3	1.08	0.11	24.4
2670	ind	2.8	1.04	0.049	37.7
	asy	2.6	1.04	0.049	40.8
3540	ind	6.2	2.05	0.057	28.5
	asy	6.1	2.05	0.056	28.0
5315	ind	8.0	1.04	0.024	8.3
	asy	7.8	1.04	0.024	8.9
6288	ind	6.2	1.02	0.047	9.4
	asy	6.1	1.02	0.047	10.0
7552	ind	4.9	1.09	0.015	6.5
	asy	4.6	1.09	0.015	7.1

Table 9.11 Fault Resolution, Scan DAPPER, $n_1 = n$

by about 10%, to good, as in circuit C7552 where the amount of simulation was reduced to 7% of its original level.

The factor which contributed most to this reduction is the ability of the first-fail counter to identify the output on which the fault was observed. For example, for a circuit with m outputs and first-fail count is r, the fault occurred on output r modulo m (where outputs are numbered from 0). Using this information, a fault which has no physical path to the observed faulty output need not be simulated at all. This information is of little benefit in circuit C1355, where many faults are observable at all outputs and most have similar detection probabilities.

These improvements are not without cost, however, since Scan DAPPER is more complex to implement than Standard DAPPER, especially when n_1 is set to n. See section 7.2 for more details. It appears that serial streaming can result in substantial reductions in post-test simulation for fault resolution in DAPPER, but the added costs in tester hardware may be prohibitive. The greatest benefits will be observed on "wide" circuits where most faults affect only a few of the many outputs, although these circuits can have the greatest overheads for Scan DAPPER for large n_1. On the other hand, the overheads are reduced using the parity block techniques described in chapter 6. Finally, the gains may be reduced if a dictionary of first failing patterns is developed before diagnosis begins (this process also gives the fault coverage of the test set).

9.3.4 Variable Test Length

These experiments are intended to show the effect of test length on the amount of post-test simulation performed, both as a percentage of a full fault simulation and as an absolute amount. As in section 9.2.4, the circuits investigated are the 74LS181 ALU and benchmark circuit C880, for test lengths set to powers of 2.

Table 9.12 gives the results for the ALU for test lengths ranging from 64 to an exhaustive test. Several points are worth noting. First, the relative amount of simulation decreases, as expected, as test length increases. Second, the absolute amount of simulation, which reflects the fact that a full fault simulation requires effort proportional to test length, decreases initially, and then increases, resulting in a minimum value for a test length of 128. Finally, this minimum value may be of limited use, since there are undetected faults for all test lengths below 512. However, the results suggest that an average test of length about 128 would be optimal for use with k-Max DAPPER.

Similar results to the ALU are observed for C880, as shown in table 9.13. The relative amount of fault simulation necessary declines from about 1.5% for the independent model and 1.3% for the asymmetric by about a factor of 10 as test length increases to 32768. The absolute amount of simulation, on the other hand, declines as test length increases to 512, but then begins to increase again. Finally, a test length of 16384 is required before all faults are detected. Again, however, k-Max DAPPER could take advantage of the simulation benefits at the shorter test lengths.

9.3.5 k-Max DAPPER

One way to reduce the increasing amount of absolute simulation noted in the previous section is to restrict test lengths to those needed to detect a fault k times, using

Test Length	Model	h_1 (%)	h_2	Rel. Effort (%)	Abs. Sim. Amnt	Undetected
64	ind	18.4	1.03	2.423	155.1	7
	asy	14.6	1.03	1.981	126.8	
128	ind	11.8	1.02	1.150	147.2	2
	asy	9.7	1.03	0.912	116.7	
256	ind	8.8	1.01	0.670	171.5	1
	asy	8.0	1.01	0.590	151.0	
512	ind	7.9	1.00	0.491	251.4	0
	asy	6.9	1.00	0.419	214.5	
1024	ind	6.8	1.00	0.435	445.4	0
	asy	5.8	1.00	0.358	366.6	
2048	ind	5.9	1.00	0.366	749.6	0
	asy	5.1	1.00	0.323	661.5	
4096	ind	5.8	1.00	0.351	1437.7	0
	asy	4.8	1.00	0.309	1265.7	
8192	ind	5.5	1.00	0.320	2621.4	0
	asy	4.5	1.00	0.284	2326.5	
16384	ind	5.3	1.00	0.303	4964.4	0
	asy	4.5	1.00	0.274	4489.2	

Table 9.12 Fault Resolution versus Test Length, 74LS181 ALU

k-max DAPPER, as described in section 7.3. In this way, regardless of the maximum test length permitted, each individual fault has its own test length – the minimum of the length needed to detect the fault *k* times, and the maximum length permitted. Experiments were performed using a maximum test length of 2048, and the results are shown in table 9.11

Two classes of effort are indicated by this table. First is that required per fault for demand-driven fault location, as a percentage of the effort required for a complete dictionary construction These results may be directly compared with those given in table 9.7 The reductions in simulation are evident Setting *k* to 10 reduces simulation over *k* being set to 20 which is a reduction over $k = 30$ However the possibility for misdiagnosis is higher for smaller *k*

The second effort column lists the simulation complexity of generating a fault dictionary for *k*-Max DAPPER as a percentage of that required for fixed-length schemes such as Standard DAPPER or signature by simulation The top figure gives the total

Test Length	Model	h_1 (%)	h_2	Rel Effort (%)	Abs Sim Amnt	Undetected
128	ind	11.9	1.26	1 469	188 0	55
	asy	10 6	1 27	1 296	165 9	
256	ind	9 5	1.18	0.739	189.2	33
	asy	8 6	1.26	0 679	173 8	
512	ind	5 8	1 09	0 339	173 6	23
	asy	5 4	1.11	0 309	158.2	
1024	ind	5.1	1.05	0.238	213.7	15
	asy	4.6	1 06	0.217	222.2	
2048	ind	4 8	1 04	0 202	113 7	9
	asy	4 4	1.06	0 184	376 8	
4096	ind	4 0	1 02	0.175	716 8	3
	asy	3.8	1 02	0 168	688 1	
8192	ind	4 0	1.03	0.165	1351 7	1
	asy	3.7	1.03	0.161	1318.9	
16384	ind	3 7	1.04	0.154	2523 1	0
	asy	3 4	1.04	0 142	2326 5	
32768	ind	3.7	1.03	0.149	4882.1	0
	asy	3 4	1.03	0 141	4620 3	

Table 9.13 Fault Resolution versus Test Length, C880

effort (e.g. 16 5% for C432 and $k = 20$), while the figure in brackets gives the effort if undetected faults are not included in the cost The top figure is an upper bound for test lengths longer than 2048. That is, if the test length was increased this relative effort would decrease. This effect was demonstrated by repeating the experiment for longer test lengths for those circuits for which 2048 does not detect all faults (C880, C1908, C2670, C3540, C5315, and C7552) The results are reported in table 9 15 Note that computing resource constraints again prevented complete test sets for C2670 (about 6 million vectors) and C7552 (more than 30 million patterns) from being investigated Nonetheless, in each case the relative effort decreased as test length increased, as expected.

9.3.6 Effect of Detection Probability Estimation

As in section 9 2 5, this experiment determines the effect on the amount of post-test simulation of the accuracy of detection probability estimates The estimated values are

Circuit	Model	h_1 (%)	h_2	Effort (demand-driven, %)			Effort (dictionary, %)		
		(k=20)	(k=20)	(k=10)	(k=20)	(k=30)	(k=10)	(k=20)	(k=30)
alu	ind	12.4	1.00	0.046	0.054	0.061	5.3	9.6	13.2
	asy	13.1	1.00	0.049	0.056	0.064	(5.3)	(9.6)	(13.2)
C432	ind	16.3	1.02	0.124	0.133	0.144	10.0	16.5	21.9
	asy	16.6	1.02	0.126	0.136	0.146	(9.3)	(15.9)	(21.3)
C499	ind	23.3	1.04	0.295	0.303	0.313	14.6	20.6	26.0
	asy	22.7	1.04	0.295	0.303	0.313	(13.7)	(19.8)	(25.2)
C880	ind	8.9	1.15	0.077	0.082	0.087	14.2	21.7	27.4
	asy	9.2	1.04	0.078	0.083	0.088	(13.2)	(20.9)	(26.6)
C1355	ind	16.2	1.14	0.335	0.346	0.359	21.8	28.3	34.2
	asy	17.0	1.13	0.339	0.351	0.364	(21.2)	(27.8)	(33.7)
C1908	ind	8.0	1.06	0.119	0.126	0.132	27.8	33.3	37.4
	asy	8.4	1.06	0.120	0.127	0.133	(26.1)	(31.7)	(35.8)
C2670	ind	7.8	1.08	0.067	0.070	0.073	26.9	32.2	36.1
	asy	7.5	1.07	0.065	0.068	0.071	(12.4)	(18.7)	(23.4)
C3540	ind	10.0	2.02	0.118	0.121	0.123	18.0	25.2	30.6
	asy	10.1	2.02	0.119	0.122	0.124	(14.1)	(21.7)	(27.3)
C5315	ind	14.0	1.07	0.217	0.223	0.228	12.5	19.1	25.4
	asy	13.9	1.06	0.216	0.222	0.227	(11.5)	(18.2)	(24.5)
C6288	ind	28.6	1.20	0.189	0.199	0.208	2.9	4.5	6.0
	asy	28.7	1.19	0.188	0.198	0.207	(2.5)	(4.1)	(5.6)
C7552	ind	10.4	1.11	0.138	0.140	0.143	19.2	26.0	31.7
	asy	10.5	1.11	0.140	0.142	0.145	(13.6)	(20.8)	(27.0)

Table 9.14 k-max simulation, 2048 maximum length

used by DAPPER to resolve individual faults on the 74LS181 ALU with a test of length 2048. The estimates, as before, are from COP and statistical techniques for lengths from 64 to 16384. The last length is an exhaustive test. The results of this experiment are shown in table 9.16.

Table 9.16 shows an average reduction of only a factor of 2 in the amount of simulation required to isolate a single fault in the ALU when the sample size for estimating detection probability increases from 64 to 16384, a 128-fold increase. Similarly, using COP results in about a 40% increase in the amount of simulation over the poorest statistical result. Hence, detection probability should be estimated using as few vectors as possible: a good heuristic might be the random test length where the "random-easy" faults have been detected. This would group easy faults by detection probability, while leaving "hard" faults together with redundant faults as estimated at detection probability 0. Yet another solution is to combine the predictions of COP with a small amount

Circuit	Length n	Missed	Effort (dictionary, %)		
			(k=10)	(k=20)	(k=30)
C880	21568	0	3.7 (3.7)	5.5 (5.5)	6.6 (6.6)
C1908	11008	0	12.7 (12.3)	17.9 (17.5)	21.4 (21.1)
C2670	50016	287	17.6 (2.5)	20.0 (5.3)	22.7 (8.5)
C3540	50016	0	4.8 (0.9)	5.4 (1.5)	6.0 (2.1)
C5315	3072	0	9.2 (8.2)	14.0 (13.1)	18.5 (17.6)
C7552	16384	387	9.6 (4.7)	11.1 (6.3)	12.4 (7.7)

Table 9.15 k-max simulation, longer test lengths

of simulation, e.g. 32 or 64 patterns, to check the results for easy faults

9.3.7 Results

It has been shown that DAPPER is able to locate a single fault in a demand-driven fashion with a small amount of effort in comparison to that required to produce a conventional signature fault dictionary. Simulation effort is reduced with more accurate estimation of detection probability, although possibly not enough to justify the increased cost. Similarly, while the relative amount of simulation decreases with increasing test length, in absolute terms it begins to increase after a certain point

Again, in most cases, the use of the asymmetric error model rather than the independent error model improves DAPPER's performance, in terms of the simulation reductions possible with it, although the differences are occasionally minimal

The use of serial streaming, rather than the parallel to serial compaction of a MINSR, can result in dramatic reductions in simulation because of the additional information available as to the failing output, but the added cost in terms of testing hardware overhead may not justify this simulation savings

Finally, the use of k-Max DAPPER to reduce simulation overhead in both demand-driven and dictionary applications has been demonstrated

Prob. Est. Len.	Model	h_1 (%)	h_2	Effort (%)
COP	ind	22.8	1.00	0.690
	asy	21.0	1.00	0.647
64 vectors	ind	13 9	1 00	0.521
	asy	12 3	1.00	0 474
128 vectors	ind	9 6	1.00	0.365
	asy	8.0	1.00	0.301
256 vectors	ind	7.8	1.00	0.304
	asy	6 5	1 00	0.268
512 vectors	ind	6.4	1.00	0 251
	asy	5.5	1.00	0 233
1024 vectors	ind	6.0	1.00	0.261
	asy	5 2	1.00	0.219
2048 vectors	ind	5.9	1.00	0 287
	asy	5.1	1 00	0.243
4096 vectors	ind	5.5	1.00	0.264
	asy	4 8	1.00	0.215
8192 vectors	ind	5.7	1 00	0 261
	asy	4.7	1.00	0.213
16384 vectors	ind	5.8	1.00	0 258
	asy	4.7	1 00	0.215

Table 9.16 Fault Resolution with Various Probability Estimates

9.4 Intermediate Signature Collection

This section outlines some experiments conducted to investigate the behaviour of Intermediate Signature Collection (ISC) (see section 4 2) The test lengths and generation method used were identical to those described in section 9 2 2 Unfortunately, the initialization conditions for the registers were slightly different so the fault coverage figures are not identical A single fault model was assumed so output storage areas were not employed Rather, faults were simulated to generate matching signatures on the failing blocks Block size was taken to be $L = 32$ bits, since that is the word length on the machines used for the experiment In the event that several faults produced the observed faulty signature, two additional faulty blocks were examined to verify the results Unfortunately, the experimental tool developed to perform the analysis was unable to deal with the amount of data generated by C7552, so no results are available for that circuit.

Circuit	Faults	Missed	Diagnosed	Incomplete	Effort (%)
alu	237	0	237	0	1.58
C432	524	0	518	2	1.68
C499	758	0	745	5	1.68
C880	942	10	923	9	1.61
C1355	1574	3	1408	155	1.81
C1908	1879	35	1650	185	1.78
C2670	2747	314	2304	12	1.63
C3540	3428	19	3239	33	1.71
C5315	5350	2	5284	5	1.61
C6288	7744	0	7710	0	1.56

Table 9.17 Performance of ISC, $n = 2048$

The experiment was repeated for every fault and the results are summarized in table 9.17. The columns are as follows: circuit name, followed by the total number of modelled faults, the number of these which were not redundant and went undetected or were aliased in the test set, the number which could be definitely diagnosed to an equivalence class, the number which neither went undetected nor were definitely diagnosed and hence were incompletely identified, and finally the average effort for each diagnosis as a percentage of the effort to generate a complete dictionary. Again, note that this effort is based on the Assumption 2.1 which equates simulation cost with the number of faults and patterns simulated. It would be unfair to compare actual CPU times when the tool developed for the ISC experiments was so unrefined.

Note that in each case the effort is greater for ISC than for the Standard DAPPER experiments reported in table 9.7. The comparison is not entirely fair, since the absolute effort for ISC will not increase with test length to the same extent as Standard DAPPER. However, in comparison with k-Max DAPPER (table 9.14), whose total effort will increase with test length at about the same rate as ISC, ISC still lags behind. Increasing L to 256 will only increase the amount of work. In addition, there is the problem of those faults which were incompletely diagnosed. These behaved identically to other faults, which may or may not have belonged to the same equivalence class. At most 3 matching blocks were observed for these, and hence definitive statements about them cannot be made.

Recall that in section 8.4.2.1 it was predicted that DAPPER would outperform ISC whenever

$$r_{av}(h_1 + kh_2) < Lh \tag{8.4.1}$$

Substituting in the values for h_1 and h_2 observed in section 9.3.5, plus 32 for L and the appropriate h values yields a maximum value for r_{av} for DAPPER to outperform ISC This value, plus that observed for the test sets are given in table 9.18.

Circuit	Observed	Maximum		
	r_{av}	$k = 10$	$k = 20$	$k = 30$
alu	17	193	154	128
C432	65	175	158	145
C499	102	130	123	117
C880	111	312	279	251
C1355	187	189	181	174
C1908	264	374	351	330
C2670	393	391	373	356
C3540	173	302	286	272
C5315	97	225	222	219
C6288	31	111	111	110

Table 9.18 Maximum and Observed values for r_{av}

Note from table 9.18 that (8.4.1) is conservative in its predictions. For example, circuits C1355 and C2670 are predicted to require about the same effort with DAPPER and ISC, yet the experimentally observed results are significantly better with DAPPER. This is because the expression uses an upper bound on the effort required with DAPPER, in that it ignores the effect of limited test length, while giving a lower bound on the complexity of ISC in that it assumes that every fault can be located after simulating only one block

The results for these circuits demonstrate that DAPPER, particularly k-Max DAPPER is able to outperform ISC when the single stuck-at fault model is used

Chapter 10 Conclusions

With the increasing complexity of VLSI circuits, testing has become a significant cost concern. Tests using pseudo-randomly generated input stimuli together with data compaction techniques to reduce output volume have been suggested as a method of reducing these costs. Such signature testing schemes have been successful in terms of fault detection, but fault location — identifying which of many possible failures is the cause of faulty behaviour — has remained a difficult problem.

This dissertation identifies a set of criteria which must be met by any fault location method for signature testing: Such a method must have

- the ability to extract location information from its signatures,
- a small fault dictionary size,
- resolution to a single fault equivalence class,
- the potential for demand-driven dictionary construction,
- applicability to multiple-output combinational circuits,
- the ability to use pseudo-random input vectors,
- no requirement to store actual output vectors, and
- a potential hardware implementation.

Each of three previously proposed methods of fault location in the RCT environment fails one or more of these criteria. The methods are signature by simulation, intermediate signature collection, and algebraic analysis. The dissertation proposes a new approach — hierarchical fault diagnosis — which meets all these criteria.

The hierarchical fault diagnosis technique consists of using several signatures, each of which contains some location information. These signatures are known as predictors and are able to reduce the potential fault set at each stage of fault location. This reduction permits demand-driven fault location to be performed with considerably less effort than might be required otherwise.

Even when fault location is performed using a fault dictionary, the proposed signatures are able to reduce overall effort significantly over previous techniques The extent of these savings has been evaluated using a tool-independent approach to fault simulation complexity and a new model for the fault coverage curve. The latter permits the relative simulation savings between the methods to be evaluated in a straightforward manner.

The improvement that hierarchical fault diagnosis represents over previous methods has been analyzed in a detailed cost comparison between them The superiority of the first has been shown over a wide range of conditions Extensive experimental verification of the performance of the hierarchical fault location technique has also been provided.

In-depth investigation of the hierarchical approach itself is provided, including:

- development of structures to implement signatures for random testing which contain useful location information.
- analysis of the tradeoffs involved between hardware overhead and diagnostic performance of diagnosis signatures, and
- detailed description of potential applications of hierarchical fault diagnosis, including expressions for expected values, variance, and aliasing probability of signatures.

The dissertation also contains other contributions to the related field of built-in self-test (BIST) including:

- a new error model for BIST: the asymmetric error model,
- a new partial matching algorithm for the output data compaction BIST technique, and
- exact and asymptotic expressions for the aliasing probability of weight counting under the independent, asymmetric, and generalized error models.

10.1 Open Problems

A number of problems remain open in this area. These are outlined below:

- Can sequential modifier blocks with guaranteed matching of a given stream be developed?
- Is it possible to develop parallel to serial compaction schemes which reduce cancellation aliasing below the levels found in MINSR and parity chain schemes used in this dissertation?
- What is the optimum number of parity chains for a given set of possibly correlated outputs?
- Can sequential circuits be adequately tested with random vectors?

- Can the results of this dissertation be extended to include diagnosis of hierarchical groupings of circuits, as in board-level testing?

Appendix A Aliasing Probability of Weight Counting

In this appendix, the aliasing probability of weight counting under a variety of error models is developed. The results are obtained for a single output stream and are used in section 7.4

A weight count with n test vectors will alias when, in the presence of a fault, equal numbers of errors occur among the output values which are 1 in the fault-free case (say w) and those which are 0 ($n - w$) in the fault-free case. Since test order is immaterial, these outputs may be considered to be grouped as shown in Figure A.1.

Figure A.1 Output regions in weight test

A description of the error models used in the following discussion is given in section 3.4.

A.1 Uniform Error Model

A weight test aliases when the weight of a faulty output stream is equivalent to that of the fault-free stream. With the uniform error model, each error stream is equally likely, so the chance of an n pattern test with fault-free output weight w aliasing is simply the total number of possible patterns of weight w (not including the fault-free one) divided by the total number of error patterns of length n, or [Sav85] [Muz87]:[*]

$$P_w^{uni} = \binom{n}{w} \frac{1}{2^n - 1} \tag{A.1.1}$$

which may be approximated, using Stirling's approximation,[†] as:

$$P_w^{uni} \sim \sqrt{\frac{w(n-w)}{2\pi n}} \left(\frac{n}{2w}\right)^{w} \left(\frac{n}{2(n-w)}\right)^{n-w} \tag{A.1.2}$$

This approximation is asymptotic as n increases with respect to w.

A.2 Independent Error Model

This section gives exact and asymptotic expressions for the aliasing probability of weight tests using the independent error model.

A.2.1 Exact Expression

The independent error model adds a parameter p, the random pattern detectability of a fault under consideration, to the uniform model.[**] Aliasing occurs when there are equal numbers of errors among the w which produce 1 at the primary output and the $n - w$ patterns which produce 0. There can be

$$\binom{w}{j}$$

different error patterns of weight j within the w patterns which produce 1, and the probability of each of these occurring is:

$$p^j (1-p)^{w-j}$$

[*] The following definition is used throughout this dissertation

$$\binom{a}{b} = \begin{cases} \frac{a!}{b!(a-b)!} & 0 \le b \le a, \\ 0 & \text{otherwise} \end{cases}$$

and $0!$ is defined to be 1.

[†] Stirling's approximation can be written as $n! \sim (2\pi n)^{\frac{1}{2}} e^{-n} n^n$ where the notation $f(n) \sim g(n)$ implies that the ratio of $f(n)$ over $g(n)$ approaches 1 as n tends to infinity.

[**] The results of this section were originally presented in [Ait88a].

Similarly, the product of the number of error patterns of weight j and their probability of occurring among the $n - w$ patterns which produce a 0 in the fault-free case is.

$$\binom{n-w}{j} \times p^j(1-p)^{n-w-j}$$

Because each output bit is assumed to be independent, these values may be multiplied together to give the probability of aliasing with $2j$ errors. Summing over all possible values of j gives the following formula for the aliasing probability of a weight test under the independent error model:

$$P_w^{ind} = \sum_{j=1}^{w}\left[\binom{w}{j}\binom{n-w}{j}p^{2j}(1-p)^{n-2j}\right] \qquad (A.2.1)$$

Some embellishments may be made to the above expression, most notably that the summation only needs to be performed until j reaches $\min(w, n-w)$, which shows that the distribution of P_w^{ind} as a function of w for a given n and p is symmetrical about $\frac{n}{2}$.

A.2.2 Asymptotic Behaviour

Equation A.2.1 bears a striking resemblance to the explicit enumeration of the *Jacobi Polynomial* (see for instance [Van87]), $P_m^{(\alpha,\beta)}(x)$, as given below.

$$P_m^{(\alpha,\beta)}(x) = 2^{-m}\sum_{j=0}^{m}\binom{m+\alpha}{j}\binom{m+\beta}{m-j}(x-1)^{m-j}(x+1)^j \qquad (A.2.2)$$

The most readily apparent difference between the two is that expression (A.2.2) includes $j = 0$ in its summation while (A.2.1) does not. The $j = 0$ term corresponds to the probability of not detecting the fault at all, P_{nd}^{ind}:

$$P_{nd}^{ind} = (1-p)^n \qquad (A.2.3)$$

By including P^{ind}_{nd} and making the following assignments

$$m = \min(w, n - w)$$

$$\alpha = n - 2m$$

$$\beta = 0$$

$$x = \frac{2p(1-p) - 1}{1 - 2p}$$

the following relationship is obtained:

$$P^{ind}_{w} = (1-p)^n \left(\frac{2}{x-1}\right)^m P^{(\alpha,\beta)}_m(x) - P^{ind}_{nd} \qquad (A.2.4)$$

which permits us to determine an asymptotic expression for P^{ind}_{w} using the method of Van Assche [Van87], which employs the *Local Limit Theorem* of Petrov [Pet75]. The full power of this theorem is not required. Instead, a simplification, as expressed in [Van87], is used:*

Theorem A.1: *(Local Limit Theorem)*: For a sequence of independent binomially distributed random variables Z_i with sequence mean μ_n and sequence variance σ_n^2,

$$\sup_N \left| \sigma_n P(Z_1 + Z_2 + \ldots + Z_n = N) - \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(N-\mu_n)^2}{2\sigma_n^2}\right) \right| \to 0$$

Proof: In Petrov, [Pet75], p. 189.

Since aliasing occurs when the difference between the number of errors among the w 1s and the $n - w$ 0s is 0, the theorem can be applied to obtain an asymptotic bound for expression A.2.1 as follows: Let $Y = \langle y_i \rangle$, $1 \le i \le n$ be the fault-free sequence, and let $X = \langle x_i \rangle$, $1 \le i \le n$ be a faulty sequence. Define each random variable Z_i, $1 \le i \le n$ as:

$$Z_i = a_i \cdot e_i$$

where

$$a_i = \begin{cases} +1, & \text{if } y_i = 0, \\ -1, & \text{otherwise} \end{cases}$$

* In fact, the version presented here is a further simplification since the [Van87] result applies to probability distributions much more general than the binomial

and

$$e_i = \begin{cases} 1, & \text{if } x_i \neq y_i; \\ 0, & \text{otherwise.} \end{cases}$$

Each e_i is the error bit at position i, and a_i represents the potential effect of an error on a weight count. Aliasing will occur when the sum of the Z_is, N, is 0. Theorem A.1 also requires the mean μ_n and variance σ_n^2 of the sum of the Z_is. Since each Z_i has probability p of being non-zero, these are given by:

$$\begin{aligned} \mu_n &= (n - 2w)p \\ \sigma_n^2 &= np(1 - p) \end{aligned}$$

Algebraic manipulation of the above equations yields the following asymptotic bound for expression (A.2.1):

$$P_w^{ind} \sim \sqrt{\frac{1}{2\pi np(1-p)}} \times \exp - \left(\frac{(n-2w)^2 p}{2n(1-p)} \right) - (1-p)^n \qquad (A.2.5)$$

which is asymptotic in that it becomes exact as n becomes large with respect to $n - 2w$

Expression (A.2.5) is invalid for $p = 0$ and $p = 1$, but its behaviour as p approaches these values may be obtained from (A.2.1):

$$\lim_{p \to 0} P_w^{ind} = 0$$

and

$$\lim_{p \to 1} P_w^{ind} = \begin{cases} 1 & w = \frac{n}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

This last result follows from the fact that $p = 1$ implies a faulty output stream which is the complement of its fault-free counterpart. An interesting corollary is that an odd test length will prevent the guaranteed aliasing which occurs when $w = \frac{n}{2}$ and $p = 1$

Figures A.2 through A.4 show the exact and estimated aliasing probability for the full range of p and several values of w for $n = 100$. The asymptotic expression of (A.2.5) is most accurate for values of p away from 0 and 1, as shown in Figures A.2 through A.4. As P_w^{ind} drops to 0 or climbs to 1, (A.2.5) matches the trend, but not always

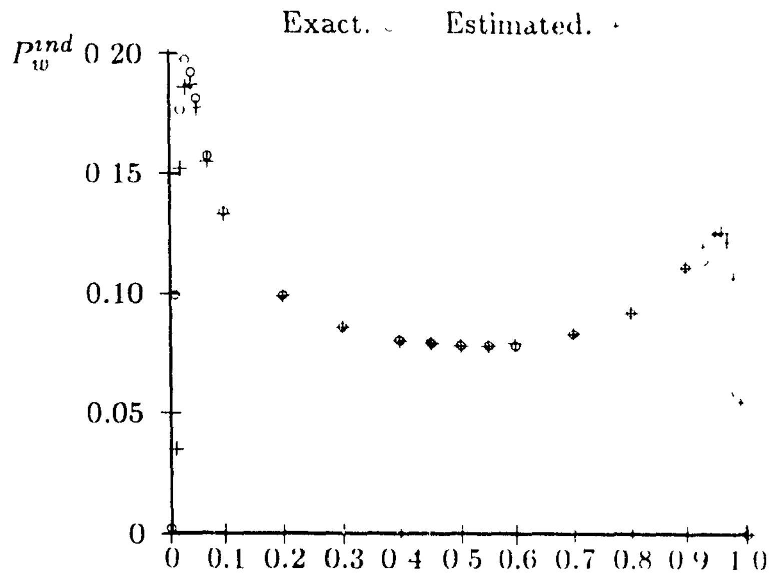


Figure A.2 Weight Aliasing Probability P_w^{ind} vs p, $n = 100$, $w = 49$ or 51

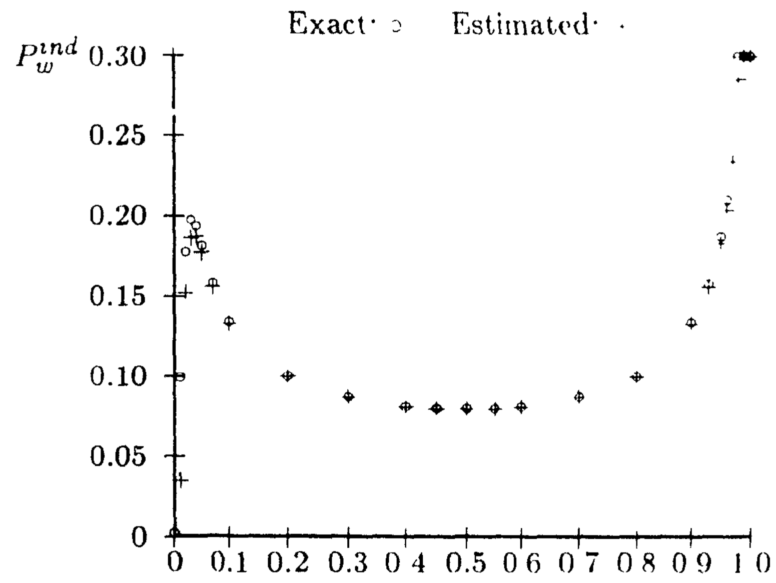


Figure A.3 Weight Aliasing Probability P_w^{ind} vs p, $n = 100$, $w = 50$

closely (see for instance the "knee" at $p = 0.9$ in Figure A.2). In addition, expression (A.2.5) can produce some absurd values of P_w^{ind} (e.g. greater than 1) when p is less than $\frac{1}{n}$. Fortunately, exact calculation of P_w^{ind} is not computationally intensive in this area, since only the first few terms of the series in (A.2.1) need to be considered when p is very small.

The asymptotic behaviour of the approximation is demonstrated by Figures A.5 through A.7. It may be seen that the approximation becomes more and more accurate

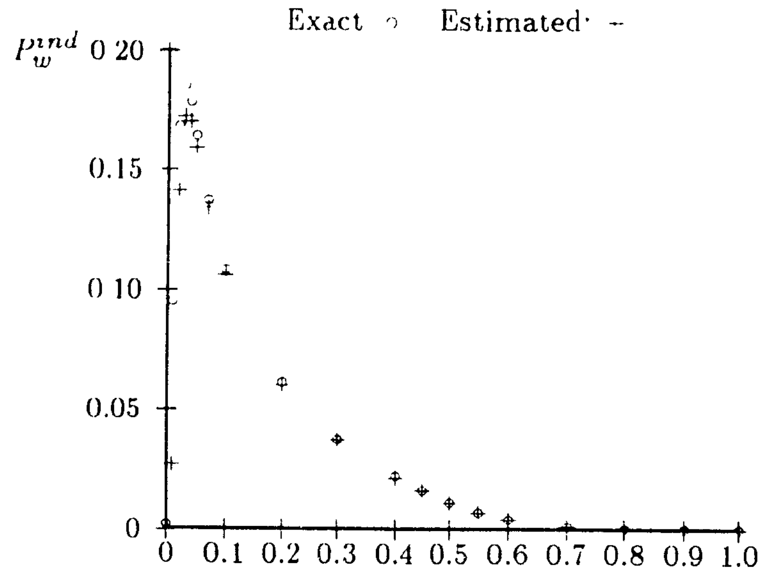


Figure A.4 Weight Aliasing Probability P_w^{ind} vs p $n = 100$, $w = 40$ or 60

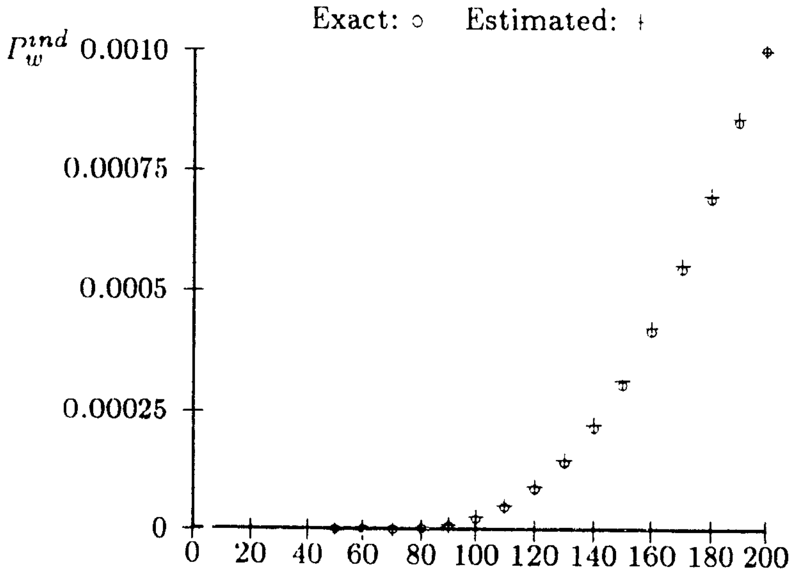


Figure A.5 Weight Aliasing Probability P_w^{ind} vs n $|n - 2w| = 40$, $p = 0.5$

with increasing n when $n - 2w$ is held constant, regardless of the value of p.

The functional behaviour of aliasing probability is shown in Figure A.8. The symmetric nature of the distribution is evident, as well as the marked discrepancies between low and high values of p Note that the distribution of aliasing versus weight is much less bell-shaped than previously believed for low values of p whereas the peak at $\frac{n}{2}$ is much more pronounced for higher p (Recall that previous works, e.g [Sav85] [Muz87], have assumed the uniform error model, where $p = 0.5$)

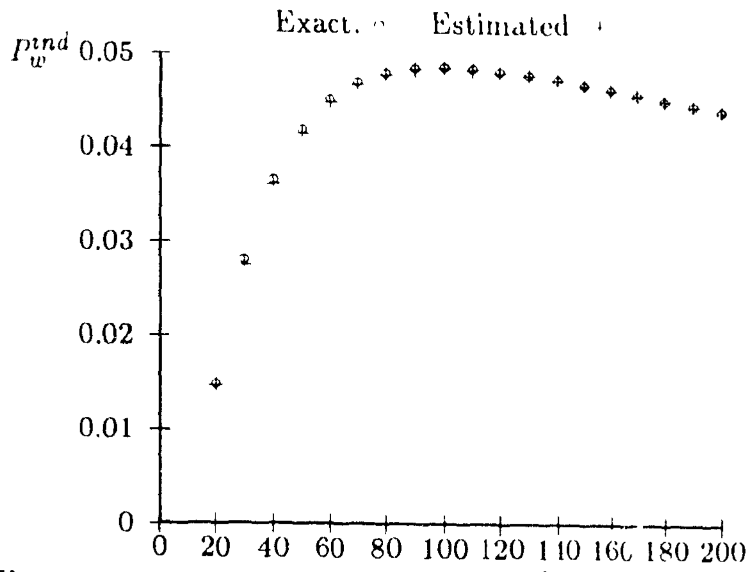


Figure A.6 Weight Aliasing Probability P_w^{ind} vs n $|n - 2w| = 10$, $p = 0.5$

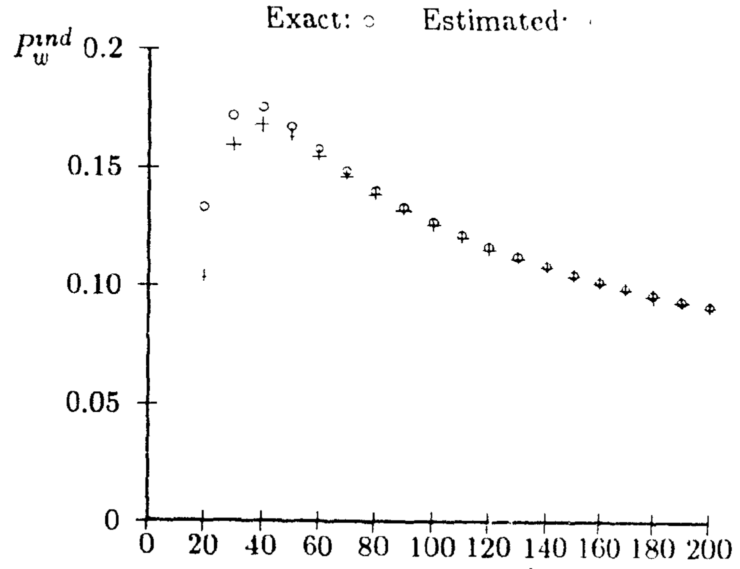


Figure A.7 Weight Aliasing Probability P_w^{ind} vs n $|n - 2w| = 10$, $p = 0.1$

A.2.3 Summary of Aliasing Behaviour

The values taken by expression (A.2.1) are not terribly obvious at first inspection, but its behaviour as n increases may be shown as follows.

Let $w = sn$ for all n and fixed s equal to the signal probability of the fault-free output function. This has the effect of keeping $\frac{n-2w}{n}$ fixed. Expression (A.2.5) may

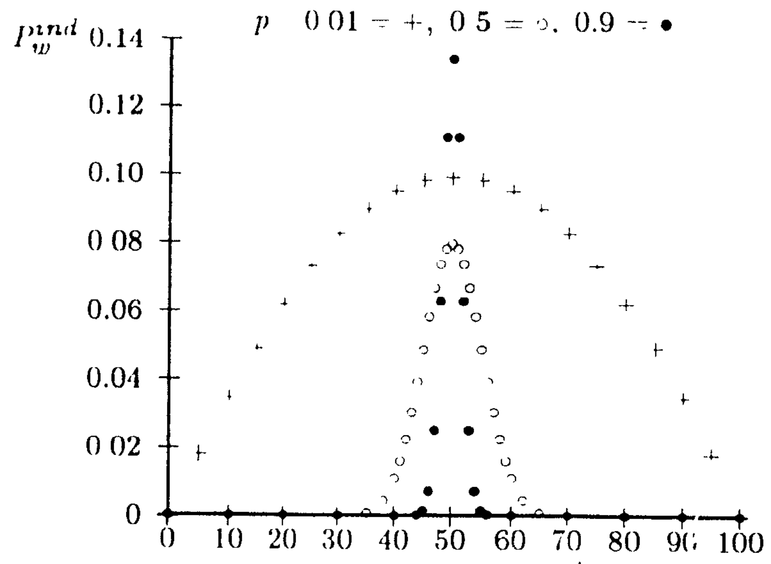


Figure A.8 Weight Aliasing Probability P_w^{ind} vs w : $n = 100$, various p

now be rewritten as:

$$P_w^{ind} \approx \sqrt{\frac{1}{2\pi np(1-p)}} \cdot \exp - \left(\frac{n(1-2s)^2 p}{2(1-p)}\right) - (1-p)^n \qquad (A.2.6)$$

Note that this expression decreases as $n^{-0.5} \times e^{-n}$, for all p, $0 < p < 1$, provided that $s \neq \frac{1}{2}$. When $s = \frac{1}{2}$, the rate of decrease is only $n^{-0.5}$.

In other words, for all functions whose inherent weight is not precisely half their length, the aliasing probability of weight counting under the independent error model decreases exponentially with increasing test length. The further the weight is from half, the faster the rate of decrease, and hence the lower the resulting aliasing probability

A.3 Asymmetric Error Model

This section gives exact and asymptotic expressions for the aliasing probability of weight tests using the asymmetric error model.

A.3.1 Exact Expression

In section A 2, the aliasing probability for weight counting under the independent error model was calculated The aliasing probability of the asymmetric error model may be derived in a similar fashion. Aliasing occurs, as before, when there are equal numbers of errors among the w patterns which produce 1 at the primary output and the $n - w$ which produce 0. There can be

$$\binom{w}{j}$$

different error patterns of weight j within the w patterns which produce 1, and the probability of each of these is:

$$(p_D)^j (1 - p_D)^{w-j}$$

Similarly, the product of the number of error patterns of weight j and their probability of occurring among the $n - w$ patterns which produce a 0 in the fault-free case is:

$$\binom{n-w}{j} \times (p_{\overline{D}})^j (1 - p_{\overline{D}})^{n-w-j}$$

Because of their assumed independence, these values may be multiplied together to give the probability of aliasing with $2j$ errors. Summing over all possible values of j gives the following formula for the aliasing probability of a weight test under the asymmetric error model:

$$P_w^{asy} = \sum_{j=1}^{\min(w,n-w)} \left[\binom{w}{j} \binom{n-w}{j} (p_D)^j (1-p_D)^{w-j} (p_{\overline{D}})^j (1-p_{\overline{D}})^{n-w-j} \right] \quad (A.3.1)$$

Note that if $p_D = p_{\overline{D}}$, expressions (A.2.1) and (A.3.1) are equivalent.

A.3.2 Asymptotic Behaviour

An asymptotic expression for (A.3.1) may be derived using Petrov's Local Limit Theorem (Theorem A.1). To use this theorem here, the two regions (see figure A.1) of the output stream are treated as independent streams of binomially distributed random variables. Aliasing will occur when each stream has the same weight, when the number of Ds minus the number of $\overline{D}$s is 0. The mean and variance of this difference of the two streams may be obtained from the laws of the binomial distribution:

$$\mu_n = wp_D - (n - w)p_{\overline{D}}$$

$$\sigma_n^2 = wp_D(1 - p_D) + (n - w)p_{\overline{D}}(1 - p_{\overline{D}})$$

In addition to these, the probability of not detecting a fault at all under the asymmetric error model, P_{nd}^{asy}, is required:

$$P_{nd}^{asy} = (1 - p_D)^w(1 - p_{\overline{D}})^{n-w} \tag{A.3.2}$$

These permit the of use Theorem A.1 in writing the following expression for asymptotic aliasing of weight testing under the asymmetric error model:

$$P_w^{asy} \sim \frac{\exp\left(\frac{-(wp_D - (n-w)p_{\overline{D}})^2}{2(wp_D(1-p_D)+(n-w)p_{\overline{D}}(1-p_{\overline{D}}))}\right)}{\sqrt{2\pi(wp_D(1 - p_D) + (n - w)p_{\overline{D}}(1 - p_{\overline{D}}))}} - (1 - p_D)^w(1 - p_{\overline{D}})^{n-w} \tag{A.3.3}$$

Note that expression (A.3.3) is invalid when both p_D and $p_{\overline{D}}$ are equal to 0 or 1. When one or both are equal to 0, aliasing probability is 0. When both are equal to 1, the independent error model applies, since $p_D = p_{\overline{D}}$, so expression (A.2.1) should be used instead.

A.3.3 Summary of Aliasing Behavior

The behaviour of expression (A.3.3) with respect to test length may be shown as follows. Since expected weight depends on test length, let $w = sn$ where s is the output signal probability. expression (A.3.3) may then be rewritten as:

$$P_w^{asy} \sim \frac{\exp\left(\frac{-n(sp_D - (1-s)p_{\overline{D}})^2}{2(sp_D(1-p_D)+(1-s)p_{\overline{D}}(1-p_{\overline{D}}))}\right)}{\sqrt{2\pi n(sp_D(1 - p_D) + (1 - s)p_{\overline{D}}(1 - p_{\overline{D}}))}} - (1 - p_D)^w(1 - p_{\overline{D}})^{n-w} \tag{A.3.4}$$

Notice that expression (A.3.4) decreases as $n^{-0.5}e^{-n}$ provided that

$$s \neq \frac{p_{\overline{D}}}{p_{\overline{D}} + p_D} \tag{A.3.5}$$

For those faults where expression (A.3.5) is false, aliasing probability decreases only as $n^{-0.5}$. For all other faults, aliasing probability decreases exponentially with test length. This result is similar to that observed with the independent error model, only in that case the worst value of s is fixed as 0.5. These latest results allow the prediction of the individual faults most likely to alias in a weight test where the asymmetric error model applies.

A.4 Generalized Error Model

The *generalized error model* allows each bit of an output sequence to have a unique error probability, say p_i for the ith such bit ($1 \le i \le n$). The asymmetric error model and hence also the independent error model are special cases of this model. The model was used in [Iva88b] in the development of aliasing probability for signature analysis. The physical motivation for the model in this context is discussed in chapter 7.

A.4.1 Exact Expression

Since each output bit has a unique error probability, p_i, the groupings of the previous sections cannot be used. Each set of errors which causes aliasing may have a unique probability of occurrence, and thus each must be summed individually in the aliasing expression.

Let $Y = \langle y_i \rangle$, $1 \le i \le n$ be the fault-free output sequence, and let Y have weight w. There are $2^n - 1$ different error streams of length n. Define some ordering on these to obtain $\langle X_j \rangle$, $1 \le j \le 2^n - 1$, where each $X_j = \langle x_{ij} \rangle$, $1 \le i \le n$. The aliasing probability of a weight test under the generalized model may then be expressed as

$$P_w^{gen} = \sum_{j=1}^{2^n-1} X_j \cdot Al(X_j) \cdot Pr(X_j) \qquad (A.4.1)$$

where

$$Al(X_j) = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} x_{ij} \ldots w, \\ 0, & \text{otherwise} \end{cases}$$

and

$$Pr(X_j) = \prod_{i=1}^{n} q_i$$

where

$$q_i = \begin{cases} p_i, & \text{if } x_{j_i} \neq y_i; \\ 1 - p_i, & \text{otherwise.} \end{cases}$$

Clearly, the generalized error model requires substantial information about the fault-free function and error sequences to produce numerical results.

A.4.2 Asymptotic Behaviour

Again, an asymptotic expression for (A.4.1) may be derived using Petrov's Local Limit Theorem (Theorem A 1). The D errors will again be subtracted from the $\overline{D}$ errors, yielding mean and variance for the difference of:

$$\mu_n = \sum_{i=1}^{n} a_i \cdot p_i$$

$$\sigma_n^2 = \sum_{i=1}^{n} p_i(1 - p_i)$$

where

$$a_i = \begin{cases} +1, & \text{if } y_i = 0; \\ -1, & \text{otherwise.} \end{cases}$$

As well, the probability of not detecting the fault at all, P_{nd}^{gen} is required:

$$P_{nd}^{gen} = \prod_{i=1}^{n}(1 - p_i) \qquad (A.4.2)$$

Using Theorem A.1, the following asymptotic expression for aliasing probability under the generalized error model is obtained:

$$P_{w}^{gen} \sim \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{\mu_n^2}{2\sigma_n^2}\right) \qquad (A.4\ 3)$$

When all of the p_is are equal to 0 or 1, expression (A 4.3) is invalid anc the exact expression should be used instead In all other cases, the rate of convergence of expression (A.4 3) depends on the nature of the individual probabilities. For this reason, the expressions of this section can be viewed more as algorithms than as equations.

References

[Abr80] M. Abramovici and M.A. Breuer, "Multiple Fault Diagnosis in Combinational Circuits Based on an Effect-Cause Analysis," *IEEE Trans. Comp* Vol C-29, pp. 451-460, June 1980.

[Abr84] M. Abramovici, P R. Menon, and D.T. Miller, "Critical Path Tracing An Alternative to Fault Simulation," *IEEE Design and Test*, February 1984

[Abr89] M. Abramovici and D.T. Miller, "Are Random Vectors Useful in Test Generation?," *Proc. ETC-89*, pp 22-25, 1989

[Aga80] V K Agarwal and A S Fung, "Predictions of Multiple Fault Coverage Capability," *Proc. FTCS-10*, pp. 307-312, 1980

[Aga81] V K Agarwal and E Cerny, "Store and Generate Built-In Testing Approach," *Proc. FTCS-11*, pp. 35-40, 1981

[Aga83] V.K. Agarwal, "Increased Effectiveness of Built-In Testing by Output Data Modification," *Proc FTCS-13*, pp 227-234, 1983

[Aga87] V K. Agarwal and Y. Zorian, "An Introduction to an Output Data Modification Scheme," in *Developments in IC Testing*, pp. 219-256, D M Miller, ed., Academic Press, London, 1987.

[Agr72] V D Agrawal and P Agrawal, "An Automatic Test Generation System for ILLIAC IV Logic Boards," *IEEE Trans Comp*, Vol. C-21, pp 1015-1017, Sept 1972.

[Agr75] P. Agrawal and V.D. Agrawal, "Probabilistic Analysis of Random Test Generation Method for Irredundant Combinational Logic Networks," *IEEE Trans Comp.*, Vol. C-24, pp. 691-695, July 1975

[Agr82] V.D. Agrawal and M.R. Mercer, "Testability Measures What Do They Tell Us?," *Proc 1982 IEEE Test Conf*, pp 391-396, 1982

[Aho74] A.V Aho, J E. Hopcroft, and J D Ullman, *The Design and Analysis of Computer Algorithms*, Addison Wesley, Reading MA, 1974

[Ait86] R.C. Aitken, *Performance Measures of Some Data Compression Techniques in Fault Detection*, M Sc Thesis, University of Victoria, Victoria, BC, April 1986.

[Ait88a] R C. Aitken and V K Agarwal, "Aliasing Probability of Non-Exhaustive Randomized Syndrome Tests," *Proc ICCAD-88*, pp 232-235, Santa Clara CA, November 1988

[Ait88b] R.C Aitken, D. Xavier, and V K. Agarwal "An Asymmetric Error Model for Built-In Self-Test," Technical Report TR-88-9R McGill University Montreal Quebec, November 1988

[Ait88c] R.C Aitken and V K Agarwal, "Error Pattern Distributions with Random and Pseudo-Random Test Vectors *Proc CCVLSI-88* pp 80-87 Halifax NS, Oct. 1988.

[Ait88d] R.C. Aitken and V K Agarwal, "Signature Coverage. Changes in BIST Performance Resulting from Known Behaviour of Certain Faults," 3rd Technical Workshop: New Directions in IC Testing, pp. 9-20, Halifax, NS, Oct 1988

[Ait89a] R.C Aitken and V K Agarwal, "A Diagnosis Method Using Pseudo-Random Vectors Without Intermediate Signatures," *Proc ICCAD-89*, pp. 574-577, Santa Clara CA, Nov. 1989.

[Ait89b] R.C Aitken, D. Xavier, A. Ivanov, and V K. Agarwal, "The Role of an Asymmetric Error Model in Predicting Aliasing of Built-In Self-Test for VLSI," *Proc CCECE-89*, pp 356-361, Montreal, PQ, Sept 1989

[Ait89c] R.C Aitken and V K Agarwal, "A Fault Location Method for Randomly Tested Sequential Circuits," 4th Technical Workshop: New Directions in IC Testing, Victoria, BC, October 1989

[Ake88] S B Akers, "A Parity Bit Signature for Exhaustive Testing," *IEEE Trans. Comp. Aided Des.*, Vol 7, pp. 333-338, March 1988.

[Ake89] S B Akers and B Krishnamurthy, "Test Counting: A Tool for VLSI Testing," *IEEE Design and Test*, Vol 6, No. 5, pp. 58-77, Oct. 1989.

[And80] H Ando, "Testing VLSI with Random Access Scan," *Proc COMPCON Spring 1980*, pp 50-52 1980

[Arm66] D B. Armstrong, "On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets," *IEEE Trans. Elec. Comp*, Vol EC-15, pp. 66-73, Feb 1966

[Arm72] D B Armstrong, "A Deductive Method for Simulating Faults in Logic Circuits," *IEEE Transactions on Computers*, Vol C-21, pp. 464-471 May 1972

[Arz81] Y Arzoumanian and J Waicukauski, "Fault Diagnosis in an LSSD Environment," *Proc. ITC-81*, pp. 86-88, Philadelphia, PA, 1981.

[Bal69] M Ball and F Hardie, "Effects and Detection of Intermittent Failures in Digital Systems," *Proc. Fall Joint Comp. Conf.*, pp. 329-335, 1969.

[Bar81] Z. Barzilai, J. Savir, G. Markowsky, and M.G. Smith, "The Weighted Syndrome Sums Approach to VLSI Testing," *IEEE Trans Comp.*, Vol. C-30, pp. 996-1000, Dec 1981.

[Bar87] P H Bardell, W H McAnney, J Savir, *Built-In Test for VLSI*, Wiley-Interscience, New York, NY, 1987.

[Bas73] D. Bastin, E. Girard, J C Rault, R. Tuillioue, "Probabilistic Test Generation Methods," *Proc. FTCS-3*, Palo Alto CA, June 1973

[Bat76] R P Batni and C.R. Kime, "A Module-Level Testing Approach for Combinational Networks," *IEEE Trans Comp.*, Vol C-25, pp. 594-604, June 1976

[Ben75] N Benowitz, D F Calhoun, G E Alderson J E Bauer C T Joeckl, "An Advanced Fault Isolation System for Digital Logic," *IEEE Trans Comp.*, Vol C-24, pp 489-497 May 1975

[Bha89] D Bhattacharya B T Murray J P Hayes "High-Level Test Generation for VLSI" *IEEE Computer*, Vol. 22, No 4 pp 16-24, April, 1989

[Bla84] T Blank, "A Survey of Hardware Accelerators Used in Computer-Aided Design," *IEEE Design and Test*, Vol 1, No. 3, pp 21-39, Aug. 1984.

[Blo82] A.J. Blodgett and D R Barbour, "Thermal Conduction Module A High-Performance Multilayer Ceramic Package," *IBM J Research and Development*, Vol 26, pp. 30-36, Jan 1982

[Bos71] D.C. Bossen and S.J. Hong, "Cause-Effect Analysis for Multiple Fault Detection in Combinational Networks," *IEEE Trans. Comp.*, Vol C-20, pp 1252-1257, Nov 1971

[Bou71] W G Bouricius, E P Hsieh, G R. Putzolu, J P Roth, P R Schneider, C J Tan, "Algorithms for Detection of Faults in Logic Circuits," *IEEE Trans Comp*, Vol. C-20, pp 1258-1264, Nov 1971

[Boz80] S. Bozorgui-Nesbat and E J McCluskey, "Structured Design for Testability to Eliminate Test Pattern Generation, *Proc FTCS-10*, pp 158-163, 1980

[Bra82] R.K. Brayton, G.D. Hachtel, L.A Memachandra, A R. Newton, A L M Sangiovanni-Vincentelli, "A Comparison of Logic Minimization Strategies Using ESPRESSO: An APL Program Package for Partitioned Logic Minimization," *Proc ISCAS-82*, pp 42-48, Rome, Italy, 1982

[Bra84] D. Brahme and J.A. Abraham, "Functional Testing of Microprocessors," *IEEE Trans. Comp*, Vol C 33, pp 475-485 June 1984

[Bre71] M A. Breuer, "A Random and an Algorithmic Technique for Fault Detection Test Generation for Sequential Circuits," *IEEE Trans Comp*, Vol C-20, pp 1364-1371, Nov. 1971.

[Brg84] F. Brglez, P. Pownall, R. Hum, "Applications of Testability Analysis From ATPG to Critical Delay Path Tracing," *Proc ITC-84*, pp 705-712, 1984

[Brg85] F. Brglez, H Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," *Proc ISCAS-85*, pp. 663 698, 1985

[Bri86] A J. Briers and K.A.E. Totton, "Random Pattern Testability by Fast Fault Simulation," *Proc ITC-86*, pp 274-281, Washington DC, Sept 1986

[CaJ82] J L. Carter, "The Theory of Signature Testing for VLSI," *Proc 14th ACM Symp. Theory Comp.*, pp. 289-296, 1982

[CaW82] W C. Carter, "The Ubiquitous Parity Bit," *Proc. FTCS-12*, pp. 289-296, June 1982

[Cha65] H Y. Chang, "An Algorithm for Selecting an Optimum Set of Diagnostic Tests," *IEEE Trans. Elec. Comp.*, Vol. EC-14, pp. 706-711, Oct. 1964.

[Cha86] S. Chakravarty and H.B. Hunt, "On the Computation of Detection Probabilities for Multiple Faults," *Proc ITC-86*, pp 252-262, Washington DC, Sept. 1986.

[Chi87] C.K. Chin and E J. McCluskey, "Test Length for Pseudorandom Testing, *IEEE Trans. Comp.*, Vol. C-36, pp 252-256, February 1987

[Cle71] F.W. Clegg, E.J. McCluskey, "Fault Equivalence in Combinational Logic Networks," *IEEE Trans Comp* Vol C-20 pp 1286-1291 Nov 1971

[Cox88] H Cox, A Ivanov, V K Agarwal and J Rajski "On Multiple Fault Coverage and Aliasing Probability Measures" *Proc ITC-88* pp 314-321 Washington DC, Sept. 1988

[Cur83] J.J. Curtin and J A Waicukauski "Multi-Chip Module Test and Diagnostic Methodology", *IBM J Research and Development* Vol 27, pp 27-34 Jan 1983.

[Dag85] M.R. Dagenais, V K. Agarwal, and N C. Rumin, "The McBoole Logic Minimizer," *Proc. 22nd Design Automation Conference*, pp 667-673, June 1985

[Dam88] M. Damiani, P. Olivo, M. Favalli, and B. Ricco, "Aliasing Errors in Signature Analysis Testing of Integrated Circuits," *Proc. ICCD-88*, pp. 458-461, Oct. 1988.

[Dam89] T.R. Damarla and M. Karpovsky, "Fault Detection in Combinational Networks by Reed-Muller Transforms," *IEEE Trans. Comp.*, Vol. 38, pp. 788-797, June 1989.

[Dav76] R. David and G. Blanchet, "About Random Fault Detection of Combinational Networks," *IEEE Trans. Comp.*, Vol. C-25, pp. 659-664, June 1976.

[Dav80] R. David, "Testing by Feedback Shift Register," *IEEE Trans. Comp.*, Vol. C-29, pp. 668-673, July 1980.

[Dav86] R. David, "Signature Analysis for Multi-Output Circuits," *IEEE Trans. Comp.*, Vol. C-35, pp. 830-837, Sept. 1986.

[Dev88] S. Devedas, H-K T. Ma, A.R. Newton, and A.L.M. Sangiovanni-Vincentelli, "Synthesis and Optimization Procedures for Fully and Easily Testable Sequential Machines," *Proc. ITC-88*, pp. 621-630, Washington DC, Sept. 1988.

[Dow64] R.W. Downing, J.S. Nowak, L.S. Tuomenoksa, "No. 1 ESS Maintenance Plan," Bell Sys. Tech. Jour., Vol 43, pp. 1961-2020, Sept. 1964.

[Eic78] E.B. Eichelberger and T.W. Williams, "A Logic Design Structure for LSI Testing," *Proc. 14th Design Automation Conf.*, pp. 462-468, June 1977.

[Eld59] R.D. Eldred, "Test Routines Based on Symbolic Logical Statements," *Journal of the ACM*, Vol. 6, pp. 33-36, 1959.

[Fel89] E. Feldman, P. Gelsinger, "Design for Test Features on the 486 Microprocessor," 12th IEEE Workshop on Design for Testability, Vail, CO, April 1989.

[Fit90] K. Fitzgerald, "Boundary Scan Standard Passes Crucial IEEE Vote," *IEEE Spectrum*, Vol. 27, No. 2, pp. 37, Feb. 1990.

[Fle89] H. Fleisher, J. Giraldi, R. Phoenix, and M. Tavel, "Minimizability of Random Boolean Functions," *IEEE Trans. Comp.*, Vol. C-38, pp. 593-595, April 1989.

[Fri67] A.D. Friedman, "Fault Detection in Redundant Circuits," *IEEE Trans. Elec. Comp.*, Vol. EC-16, pp. 99-100, Feb. 1967.

[Fro77] R.A. Frohwerk, "Signature Analysis: A New Digital Field Service Method," *Hewlett-Packard Journal*, pp. 2-8, May 1977.

[Fuj82] H. Fujiwara and S. Toida, "The Complexity of Fault Detection Problems in Combinational Logic Circuits," *IEEE Trans. Comp.*, Vol. C-31, pp. 555-560, June 1982.

[Fuj83] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms," *IEEE Trans. Comp.*, Vol. C-32, pp. 1137-1144, Dec. 1983.

[Fuj85] H. Fujiwara, *Logic Testing and Design for Testability*, The MIT Press, Cambridge MA, 1985.

[Fun75] S. Funatsu, N. Wakatsuki, and T. Arima, "Test Generation Systems in Japan," *Proc. 12th Design Automation Symp.*, pp. 114-122, June 1975.

[Fun89] S. Funatsu, M. Kawai, A. Yamada, "Scan Design at NEC," *IEEE Design and Test*, Vol. 6, No. 3, pp. 50-57, June 1989.

[Gae86] R.K. Gaede, M.R. Mercer, and B. Underwood, "Calculation of Greatest Lower Bounds Obtainable by the Cutting Algorithm," *Proc. ITC-86*, pp. 498-505, Washington DC, Sept. 1986.

[Gal80] J. Galiay, Y. Crouzet, and M. Vergniault, "Physical vs. Logical Fault Models of MOS LSI Circuits: Impact on Their Testability," *IEEE Trans. Comp.*, Vol. C-29, pp. 527-531, June 1980.

[Gar78] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., New York, NY, 1978.

[Gel86] P. Gelsinger, "Built-In Self-Test of the 80386," *Proc. ICCD-86*, pp. 169-173, Oct. 1986.

[Glo88] C.S. Gloster and F. Brglez, "Boundary Scan with Cellular-Based Built-In Self-Test," *Proc. ITC-88*, pp. 138-145, Washington DC, Sept. 1988.

[Goe80] P. Goel, "Test Generation Costs: Analysis and Projections," *Proc. 17th Des. Aut. Conf.*, pp. 77-84, June 1980.

[Goe81] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Trans. Comp.*, Vol. C-30, pp. 215-222, Mar. 1981.

[Gol79] L.H. Goldstein, "Controllability/Observability Analysis of Digital Circuits," *IEEE Trans. Circuits and Systems*, Vol. CAS-26, pp. 685-693, Sept. 1979.

[Gol67] S.W. Golomb, *Shift Register Sequences*, Holden-Day Inc., San Francisco CA, 1967.

[Gra89] D. Graham, *Introduction to Multi-Strategy Testing*, GenRad Inc., Concord MA, 1989.

[Gup88] S.K. Gupta and D.K. Pradhan, "A Framework for Designing and Analyzing BIST Techniques," *Proc. ITC-88,* pp. 329-342, Washington, DC, Sept. 1988.

[Har87] D. Harel and B. Krishnamurthy, "Is There Hope for Linear Time Fault Simulation?," *Proc. FTCS-17*, pp. 28-33, July 1987.

[Har89] W. Harwood and M. McDermott, "Testability Features of the MC68332 Modular Microcontroller," *Proc. ITC-89,* pp. 615-623, Washington, DC, Aug. 1989.

[Has84] S.Z. Hassan and E.J. McCluskey, "Increased Fault Coverage Through Multiple Signatures," *Proc. FTCS-14*, pp. 354-358, June 1984.

[Has88] A. Hassan, J. Rajski, and V.K. Agarwal, "Testing and Diagnosis of Interconnects Using Boundary Scan Architecture," *Proc. ITC-88*, pp. 126-137, Washington DC, Sept. 1988.

[Hay71] J.P. Hayes, "A NAND Model for Fault Diagnosis in Combinational Logic Networks," *IEEE Trans. Comp.*, Vol. C-20, pp. 1496-1506, Dec. 1971.

[Hay76a] J.P. Hayes, "Transition Count Testing of Combinational Logic Circuits," *IEEE Trans. Comp.*, Vol. C-25, pp. 613-620. June 1976.

[Hay76b] J.P. Hayes, "Check Sum Methods for Test Data Compression," *J. Design Automation and Fault Tolerant Computing*, Vol. 1, pp. 3-17, June 1976.

[Hay78] J.P. Hayes, "Generation of Optimal Transition Count Testing," *IEEE Trans. Comp.*, Vol. C-27, pp. 36-41, Jan. 1978.

[Hei87] K.D. Heidtmann, "Arithmetic Spectrum Applied to Stuck-At Fault Detection for Combinational Networks," 2nd Technical Workshop: New Directions for IC Testing, paper 4, Winnipeg MB, April 1987.

[Hor88] P.D. Hortensius, and R.D. McLeod, "Cellular Automata-Based Signature Analysis for Built-In Self-Test," 3rd Technical Workshop: New Directions for IC Testing, pp. 117-128, Halifax NS, Oct. 1988.

[Hor89] P.D. Hortensius, R.D. McLeod, W. Pries, D.M. Miller, and H.C. Card, "Cellular Automata-Based Pseudorandom Number Generators for Built-In Self-Test," *IEEE Trans. Comp. Aided Design*, Vol. 8, pp. 842-859, Aug. 1989.

[Hsi84] T.C. Hsiao and S.C. Seth, "An Analysis of the Use of Rademacher-Walsh Spectrum in Compact Testing," *IEEE Trans. Comp.*, Vol. C-33, pp. 934-937, Oct. 1984.

[Hui88] L.M Huisman, "The Reliability of Approximate Testability Measures," *IEEE Design and Test*, Vol. 5, pp. 57-67, Dec. 1988.

[Hur85] S.L. Hurst, D.M. Miller, and J.C. Muzio, *Spectral Techniques in Digital Logic*, Academic Press, London, 1985.

[Hur87] S.L. Hurst, "The Interrelationships Between Fault Signatures Based on Counting Techniques," in *Developments in IC Testing*, pp. 83-114, D.M. Miller, ed., Academic Press, London, 1987.

[Hur88] S.L. Hurst, "A Hardware Consideration of CALBO Testing," 3rd Technical Workshop: New Directions for IC Testing, pp. 129-146, Halifax NS, Oct. 1988.

[Iba75] O.H. Ibarra and S.K. Sahni, "Polynomially Complete Fault Detection Problems," *IEEE Trans. Comp.*, Vol. C-24, pp. 242-249, Mar. 1975.

[Iva88a] A. Ivanov and V.K. Agarwal, "An Iterative Technique for Calculating Aliasing Probability of Linear Feedback Shift Registers," *Proc. FTCS-18*, pp. 70-75, Tokyo, Japan, June 1988.

[Iva88b] A. Ivanov, *BIST Signature Analysis: Analytical Techniques for Computing the Probability of Aliasing*, Ph.D. Dissertation, Dept. of Elect. Eng., McGill University, Montreal, PQ, 1988.

[Jac86] J. Jacob, N.N. Biswas, "GTBD Faults and Lower Bounds on Multiple Fault Coverage of Single Fault Test Sets", *Proc. ITC-86*, pp. 849-855, Washington, DC, Sept. 1986.

[Jai84] S.K. Jain and V.D. Agarwal, "STAFAN: An Alternative to Fault Simulation," *Proc. 21st Design Automation Conf.*, pp. 18-23, 1984.

[Kam75] S. Kamal, "An Approach to the Diagnosis of Intermittent Faults," *IEEE Trans. Comp.*, Vol. C-24, pp. 461-467, May 1975.

[Kar85] M. Karpovsky, ed. *Spectral Techniques and Fault Detection*, Academic Press, London, 1985.

[Kau68] W.H. Kautz, "Fault Testing and Diagnosis in Combinational Digital Circuits," *IEEE Trans. Comp.*, Vol. C-17. pp. 352-366. April 1968.

[Kim88] K. Kim, D.S. Ha, J.G. Tront, "On Using Signature Registers as Pseudorandom Pattern Generators in Built-In Self-Test," *IEEE Trans. on Comp. Aided Design*, Vol. 7, pp. 919-928. Aug. 1988.

[Koe79] B. Koenemann, J. Mucha, and G. Zwiehoff, "Built-In Logic Block Observation Techniques," *Proc. 1979 IEEE Test Conf.*, pp. 37-41. Cherry Hill NJ, 1979.

[Kra87] A. Krasniewski and S. Pilarski, "Circular Self-Test Path: A Low-Cost BIST Technique," *Proc 24th Design Automation Conf.*, pp. 407-415, 1987.

[Kri86] B. Krishnamurthy and I.G. Tollis, "Improved Techniques for Estimating Signal Probabilities," *Proc. ITC-86,* Washington, DC, Sept. 1986.

[Kub83] J.R. Kuban and W.C. Bruce, "The MC6804P2 Built-In Self-Test," *Proc. ITC-83,* pp. 295-300, 1983.

[Kum81] S.K. Kumar and M.A. Breuer, "Probabilistic Aspects of Boolean Switching Functions via a New Transform," *Jour. ACM,* Vol. 28, pp. 502-520, 1981.

[Lal85] P.K. Lala, *Fault Tolerant and Fault Testable Hardware Design,* Prentice-Hall Int., Englewood Cliffs NJ, 1985.

[Lee89] W.T. Lee, "Engineering a Device for Electron-Beam Probing," *IEEE Design and Test,* Vol. 6, No. 3, pp. 36-49, June 1989.

[Los78] J. Losq, "Efficiency of Random Compact Testing," *IEEE Trans. Comp.,* Vol. C-27, pp. 516-525, June 1978.

[Maa88] F. Maamari and J. Rajski, "A Fault Simulation Method Based on Stem Regions," *Proc. ICCAD 1988,* pp. 170-173, Santa Clara, CA, Nov. 1988.

[Maa90] F. Maamari and J. Rajski, "A Method of Fault Simulation Based on Stem Regions," *IEEE Trans. Comp. Aided Des.,* Vol 9, pp. 212-220, Feb. 1990.

[Mal88] W. Maly and P. Nigh, "Built-In Current Testing – Feasibility Study," *Proc. ICCAD 1988,* pp. 340-343, Santa Clara CA, Nov. 1988.

[Man64] D. Mandelbaum, "A Measure of Efficiency of Diagnostic Tests Upon Sequential Logic," *IEEE Trans. Elec. Comp.,* Vol. EC-13, p. 630, Oct. 1964.

[Man79] M.M. Mano, *Digital Logic and Computer Design,* Prentice-Hall, Englewood Cliffs NJ, 1979.

[Mar81] G. Markowsky, "Syndrome-Testability Can be Achieved by Circuit Modification," *IEEE Trans. Comp.,* Vol. C-30, pp. 604-608, Aug. 1981.

[Mau87] C. Maunder and F. Beenker, "Boundary Scan, A Framework for Structured Design-For-Test," *Proc. ITC-87,* pp. 724-729, Washington DC, Sept. 1987.

[Max89] P.C. Maxwell, "A Comparative Study of Test Sets for PLAs," 12th IEEE Workshop on Design for Testability, Vail CO, April 1989.

[McA87] W.H. McAnney and J. Savir, "There is Information in Faulty Signatures," *Proc ITC-87,* pp. 630-636, Washington DC, Sept. 1987.

[McC56] E.J. McCluskey, "Minimization of Boolean Functions," *Bell Sys. Tech. Jour.,* Vol. 35, pp. 1417-1444, Nov. 1956.

[McC81] E.J. McCluskey and S. Bozorgui-Nesbat, "Design for Autonomous Test," *IEEE Trans. on Circuits and Systems,* Vol. CAS-28, pp. 1070-1079, Nov. 1981.

[McC85] E.J. McCluskey, "Built-In Self-Test Techniques," *IEEE Design and Test,* Vol. 2, No. 2, pp. 21-28, April 1985.

[McC86] E.J. McCluskey, *Logic Design Principles with Emphasis on Testable Semi-Custom Circuits.* Prentice-Hall. Englewood Cliffs NJ. 1986.

[McC87] E.J. McCluskey and S. Mourad. "Comparing Causes of IC Failure." in *Developments in IC Testing,* D.M. Miller, ed., pp. 13-46, Academic Press, London 1987.

[Mea80] C. Mead and L. Conway, *Introduction to VLSI Systems,* Wiley, New York NY, 1980.

[Mei74] K.C.Y. Mei, "Bridging and Stuck-at Faults," *IEEE Trans. Comp.,* Vol. C-23, pp. 720-727, July 1974.

[Mer89] M.R. Mercer, comment at 12th IEEE Design for Testability Workshop, Vail CO, April 1989.

[Mil64] F. Mileto and G. Putzolu, "Average Values of Quantities Appearing in Boolean Function Minimization," *IEEE Trans. Elec. Comp.*, Vol. EC-13, pp. 87-92, April 1964.

[Mil83] D.M. Miller and J.C. Muzio, "Spectral Fault Signatures for Internally Unate Combinational Networks," *IEEE Trans. Comp.*, Vol. C-32, pp. 1058-1062, Nov. 1983.

[Mil84] D.M. Miller and J.C. Muzio, "Spectral Fault Signatures for Single Stuck-at Faults in Combinational Networks," *IEEE Trans. Comp.*, Vol. C-33, pp. 765-769, August 1984.

[Mil89] D.M. Miller and S. Zhang, "Aliasing in Multiple-Input Data Compactors," *Proc. CCECE-89*, pp. 347-351, Montréal, PQ, Sept. 1989.

[MuE90] E.E. Murphy, "Applications '90: Design Tools," *IEEE Spectrum*, Vol. 27, No. 2, pp. 34-36, Feb. 1990.

[MuF90] F. Muradali, V.K. Agarwal and B. Nadeau-Dostie, "A New Procedure for Weighted Random Built-In Self-Test," submitted to ITC-90.

[Muz87] J.C. Muzio, F. Ruskey, R.C. Aitken, M. Serra, "Aliasing Probabilities for Some Data Compression Techniques," in *Developments in IC Testing*, D.M. Miller, ed., pp. 169-217, Academic Press, London 1987.

[Muz88] J.C. Muzio, "Concerning Parity Bit Signatures for Testing," 3rd Technical Workshop: New Directions for IC Testing, pp. 157-167, Halifax NS, Oct. 1988.

[Nig90] P. Nigh and W. Maly, "Test Generation for Current Testing," *IEEE Design and Test*, Vol. 7, No. 2, pp. 26-38, Feb. 1990.

[Par75a] K.P. Parker and E.J. McCluskey, "Analysis of Logic Circuits with Faults using Input Signal Probabilities," *IEEE Trans. Comp.*, Vol. C-24, pp. 573-578, May 1975.

[Par75b] K.P. Parker and E.J. McCluskey, "Probabilistic Treatment of General Combinational Networks," *IEEE Trans. Comp.*, Vol. C-24, pp. 668-670, June 1975.

[Par87] K.P. Parker, *Integrating Design and Test: Using CAE Tools for ATE Programming*, IEEE Computer Society Press, New York NY, 1987.

[Pat90] S. Pateras, "Testability Measures - A Review," part of Ph.D. Dissertation, Dept. of Elect. Eng., McGill University, in preparation.

[Per89] T.S. Perry, "Intel's Secret is Out," *IEEE Spectrum*, Vol. 26, No. 4, pp. 22-28, April 1989.

[Pet72] W. W. Peterson and E. J. Weldon. *Error Correcting Codes*. 2nd ed., MIT Press, Cambridge MA. 1972.

[Pet75] V.V. Petrov, *Sums of Independent Random Variables*, (English translation), Springer-Verlag, Berlin 1975.

[Pra88] M.M. Pradhan. E.J. O'Brien. S.L. Lam. J. Beausang. "Circular BIST with Partial Scan," *Proc. ITC-88*, pp. 719-729. Washington DC, Sept. 1984.

[Raj85] J. Rajski, J. Tyszer. "Combinatorial Approach to Multiple Contact Faults Coverage in Programmable Logic Arrays". *IEEE Trans. on Comp.*, Vol. C-34, pp. 549-553, June, 1985.

[Raj87] J. Rajski and H. Cox, "A Method of Test Generation and Fault Diagnosis in Very Large Combinational Circuits," *Proc. ITC-87*, pp. 932-943, Washington DC, Sept. 1987.

[Raj90] J. Rajski and H. Cox, "A Method of Generating Necessary Assignments in Algorithmic Test Pattern Generation," submitted to ITC-90.

[Rav81] K.W. Ravi, *Imperfections and Impurities in Semiconductor Silicon*, John Wiley and Sons, New York, NY, 1981.

[Red77] S.M. Reddy, "A Note on Testing Logic Circuit Transition Counting," *IEEE Trans. Comp.*, Vol. C-26, pp. 313-314, March 1977.

[Rob87] J.P. Robinson and N.R. Saxena, "A Unified View of Test Compression Methods," *IEEE Trans. Comp.*, Vol. C-36, pp. 94-99, Jan. 1987.

[Rob88] J.P. Robinson and N.R. Saxena, "Simultaneous Signature and Syndrome Compression," *IEEE Trans. Comp. Aided Des.*, Vol. 7, No. 5, May 1988.

[Rot66] J.P. Roth, "Diagnosis of Automata Failures: A Calculus and a Method." *IBM J. Res. Devel.*, Vol. 10, pp. 278-281, July 1966.

[San60] P.J. Sandiford, "A New Binomial Approximation for Use in Sampling from Finite Populations," *Amer. Stat. Assoc. Jour.*, Vol. 55, No. 12, Dec. 1960.

[Sav80a] J. Savir, "Syndrome-Testable Design of Combinational Circuits," *IEEE Trans. Comp.*, Vol. C-29, pp. 442-451, June 1980.

[Sav80b] J. Savir, "Detection of Single Intermittent Faults in Sequential Circuits," *IEEE Trans. Comp.*, Vol. C-29, pp. 673-678, July 1980.

[Sav81] J. Savir, "Syndrome Testable Design of Syndrome Untestable Circuits," *IEEE Trans. Comp.*, Vol. C-30, pp. 606-609, Aug. 1981.

[Sav83] J. Savir, "Good Controllability and Observability Do Not Guarantee Good Testability," *IEEE Trans. Comp.*, Vol. C-32, pp. 1198-1200, Dec. 1983.

[Sav84] J. Savir, G.S. Ditlow and P.H. Bardell, "Random Pattern Testability," *IEEE Trans. Comp.*, Vol. C-33, pp. 79-90, Jan. 1984.

[Sav85] J. Savir and W.H. McAnney, "On the Masking Probability with One's Count and Transition Count," *Proc. ICCAD-85*, pp. 111-113, Santa Clara, CA, November 1985.

[Sav86] J. Savir and W.H. McAnney, "Random Pattern Testability of Delay Faults," *Proc. ITC-86*, pp. 263-273, Washington DC, Sept. 1986.

[Sav88] J. Savir and W.H. McAnney, "Identification of Failing Tests with Cycling Registers," *Proc. ITC-88*, pp. 322-328, Washington, DC, Sept. 1988.

[Sax86] Saxena, N.R., and Robinson, J.P., "Accumulator Compression Testing," *IEEE Trans. Comp.*, Vol. C-35, No. 4, April 1986.

[Sax87] N.R. Saxena, "Effectiveness Measures for Data Compression Techniques Under Unequally Likely Errors," in *Developments in IC Testing*, pp. 257-278, D.M. Miller, ed., Academic Press, London 1987.

[Sch75] H.D. Schnurmann, E. Lindbloom, R.G. Carpenter, "The Weighted Random Test Pattern Generator," *IEEE Trans. Comp.*, Vol. C-24, pp. 695-700, July 1975.

[Sch88] M.H. Schulz and E. Auth, "Advanced Automatic Test Pattern Generation and Redundancy Identification," *Proc. FTCS-18*, pp. 30-35, Tokyo, Japan, June 1988.

[Sed79] R.M. Sedmak, "Design for Self-Verification: An Approach for Dealing with Testability Problems in VLSI-Based Designs," *Proc. IEEE Test Conf.*, pp. 112-124, Cherry Hill NJ, Oct. 1979.

[Sel68] F.F. Sellers, M.Y. Hsiao, and L.W. Bearnson, "Analyzing Errors with the Boolean Difference," *IEEE Trans. Comp.*, Vol C-17, pp. 676-683, July 1968

[Ser87] M. Serra and J.C. Muzio, "Testing Programmable Logic Arrays by Sum of Syndromes," *IEEE Trans. Comp.*, Vol. C-36, pp. 1097-1100, Sept. 1987.

[Ser88] M. Serra, D.M. Miller, J.C. Muzio, "Linear Cellular Automata and LFSRs are Isomorphic," 3rd. Technical Workshop, New Directions for IC Testing, pp. 213-223, Halifax NS, Oct. 1988.

[Ser89] M. Serra, University of Victoria, private communication, July 1989.

[Ses62] S. Seshu and D.N. Freeman, "The Diagnosis of Asynchronous Sequential Switching Systems," *IEEE Trans. Elec. Comp.*, Vol. EC-11, pp. 459-465, Aug. 1962.

[Ses65] S. Seshu, "On an Improved Diagnosis Program," *IEEE Trans. Elec. Comp.*, Vol. EC-14, pp. 76-79, Feb. 1965.

[Set85] S.C. Seth, L. Pan and V.D. Agrawal, "PREDICT: Probabilistic Estimation of Digital Circuit Testability," *Proc. FTCS-15*, pp. 220-225, 1985.

[Set86] S.C. Seth, B.B. Bhattacharya and V.D. Agrawal, "An Exact Analysis for Efficient Computation of Random Pattern Testability," *Proc. FTCS-16*, pp. 318-323, Vienna, July 1986.

[Sha48] C.E. Shannon, "A Mathematical Theory of Communication," *Bell Sys. Tech. Jour.*, Vol. 27, pp. 379-432, 623-656, 1948.

[She77] J.J. Shedletsky, "Random Testing: Practicality vs. Verified Effectiveness," *Proc. FTCS-7*, pp. 175-179, 1977.

[She85] J.P. Shen, W. Maly, and F.J. Ferguson, "Inductive Fault Analysis of MOS Integrated Circuits," *IEEE Design and Test*, Vol. 2, No. 6, pp. 13-36, Dec. 1985.

[Smi79] J.E. Smith, "Detection of Faults in Programmable Logic Arrays," *IEEE Trans. Comp.*, Vol. C-28, pp. 845-852, Nov. 1979.

[Smi80] J.E. Smith, "Measures of the Effectiveness of Fault Signature Analysis," *IEEE Trans. Comp.*, Vol. C-29, pp. 510-514, June 1980.

[Ste77] J.H. Stewart, "Future Testing of Large LSI Circuit Cards," *Proc. 1977 Semiconductor Test Symp.*, pp. 6-15, Oct. 1977.

[Sun84] Z. Sun and L.-T. Wang, "Self-Testing of Embedded RAMs," *Proc. ITC-84*, pp. 148-156, Oct. 1984.

[Sus73] A.K. Susskind, "Diagnostics for Logic Networks," *IEEE Spectrum*, Vol. 10, pp. 40-47, Oct. 1973.

[Sus81] A.K. Susskind, "Testing by Verifying Walsh Coefficients," *Proc. FTCS-11*, pp. 206-208, 1981.

[Swa89] G. Swan, Y. Trivedi, D. Wharton, "CrossCheck - A Practical Solution for ASIC Testability," *Proc. ITC-89*, pp. 903-908, Washington DC, Aug. 1989.

[Tot88] K. Totton and S. Shaw, "Self-Test: The Solution to the VLSI Test Problem?," *IEE Proceedings*, Vol. 135, Pt. E, No. 4, pp. 190-195, July 1988.

[Tre85] R. Treuer, H. Fujiwara, and V.K. Agarwal, "Implementing a Built-In Self-Test PLA Design," *IEEE Design and Test*, Vol. 2, No. 2, pp. 37-48, April 1985.

[Tsi62] S.H. Tsiang and W. Ulrich, "Automatic Trouble Diagnosis of Complex Logic Circuits," *Bell Sys. Tech. Jour.*, Vol. 41, p. 1177, July 1962.

[Tur89] J. Turino, "Advantages of a Dual Port Testability Bus," 12th IEEE Workshop on Design for Testability, Vail CO, April 1989.

[Ude88] J.G. Udell, "Reconfigurable Hardware for Pseudo-Exhaustive Test," *Proc. ITC-88*, pp. 522-530, Washington DC, Sept. 1988.

[Ulr74] E.G. Ulrich and T. Baker, "Concurrent Simulation of Nearly Identical Digital Networks," *IEEE Computer*, Vol. 7, pp. 39-44, April 1974.

[VaJ89] J. Vasudevamurthy and J. Rajski, "Test Generation and Synthesis of Fully Testable Multi-Level Networks," 4th Technical Workshop: New Directions for IC Testing, Victoria, BC, Oct. 1989.

[Van87] W. Van Assche, *Asymptotics for Orthogonal Polynomials,* No. 1265 in Lecture Notes in Mathematics series, Springer-Verlag, Berlin, 1987.

[VaN89] N. Vasanthavada and N. Kanopoulos, "A Built-In Test Module for Fault Isolation," *IEEE Design and Test*, Vol. 6, No. 3, pp. 58-65, June 1989.

[Wad78a] R.L. Wadsack, "Fault Modelling and Logic Simulation of CMOS and MOS Integrated Circuits," *Bell Sys. Tech. Jour.*, Vol. 57, pp. 1449-1474, 1978.

[Wad78b] R.L. Wadsack, "Fault Coverage in Digital Integrated Circuits," *Bell Sys. Tech. Jour.*, Vol. 57, pp. 1475-1488, 1978.

[Wag87] K.D. Wagner, C.K. Chin, E.J. McCluskey, "Pseudorandom Testing," *IEEE Trans. Comp.*, Vol. C-36, pp. 332-343, March 1987.

[Wai85] J.A. Waicukauski, E.B. Eichelberger, D.O. Forlenza, E. Lindbloom, and T. McCarthy, "A Statistical Calculation of Fault Detection Probabilities by Fast Fault Simulation," *Proc. ITC-85*, Philadelphia, PA, 1985.

[Wai87] J.A. Waicukauski, V.P. Gupta, and S.T. Patel, "Diagnosis of BIST Failures by PPSFP Simulation," *Proc. ITC-87*, pp. 480-484, Washington DC, Sept. 1987.

[Wai88] J.A. Waicukauski and E. Lindbloom, "Fault Detection Effectiveness of Weighted Random Patterns," *Proc. ITC-88*, pp. 245-255, Washington DC, Sept. 1988.

[Wai89] J.A. Waicukauski and E. Lindbloom, "Failure Diagnosis of Structured VLSI," *IEEE Design and Test*, Vol. 6, No. 4, pp. 49-60, Aug. 1989.

[Wan86] L.T. Wang and E.J. McCluskey, "A Hybrid Design of Maximum-Length Sequence Generators." *Proc. ITC-86*, pp. 25-37, Washington DC, Sept. 1986.

[Wes85] N. Weste and K. Eshragian. *Principles of CMOS VLSI Design: A Systems Perspective*, Addison Wesley. Reading MA. 1985

[Wil73] M.J.C. Williams and J.B. Angell, "Enhancing Testability of Large-Scale Integrated Circuits via Test Points and Additional Logic," *IEEE Trans. Comp.*, Vol. C-22, pp. 46-60. Jan. 1973.

[Wil79] T.W. Williams and K.P. Parker, "Testing Logic Networks and Designing for Testability," *IEEE Computer*, Vol. 12, No. 10, pp. 9-21, Oct. 1979.

[Wil81] T.W. Williams and N.C. Brown, "Defect Level as a Function of Fault Coverage," *IEEE Trans. Comp.*, Vol. C-30, pp. 987-988, Dec. 1981.

[Wil83] T.W. Williams and K.P. Parker, "Design for Testability — A Survey," *Proc. IEEE*, Vol. 71, pp. 98-112, Jan. 1983.

[Wil86] T.W. Williams, W. Daehn, M. Gruetzner, C.W. Starke, "Comparison of Aliasing Errors for Primitive and Non-Primitive Polynomials," *Proc. ITC 1986,*, pp. 282-288, Washington, DC, Sept. 1986.

[Wil88] T.W. Williams, W. Daehn, M. Gruetzner, C.W. Starke, "Bounds on Aliasing for Linear Feedback Shift Registers," *IEEE Trans. Comp. Aided Design*, Vol. 7, No. 1, Jan. 1988.

[Wil89] T.W. Williams and W. Daehn, "Aliasing Probability for Multiple Input Signature Analyzers with Dependent Inputs," 12th IEEE Workshop on Design for Testability, Vail CO, April 1989.

[Wol83] S. Wolfram, "Statistical Mechanics of Cellular Automata," *Rev. of Modern Phys.*, Vol. 55, pp. 601-644, 1983.

[Won77] T.H. Wonnacott and R.J. Wonnacott, *Introductory Statistics, 3rd ed.*, John Wiley and Sons, New York, NY, 1977.

[Wun87] H.-J. Wunderlich, "On Computing Optimized Input Probabilities for Random Tests," *Proc. 24th Design Automation Conference*, pp. 392-398, June 1987.

[Xav89] D. Xavier, R.C. Aitken, A. Ivanov and V.K. Agarwal, "Experiments on Aliasing in Signature Analysis Registers," *Proc. ITC-89*, pp. 344-354, Washington, DC, Aug. 1989.

[Yan87] T. Yano and H. Okamoto, "Fast Fault Diagnostic Method Using Fault Dictionary for Electron Beam Tester," *Proc. ITC-87*, pp. 561-565, Washington, DC, Sept. 1987.

[Yen69] Y.T. Yen, "Intermittent Failure Problems of Four-Phase MOS Circuits," *J. Solid-State*, Vol SC-4, pp. 107-110, 1969.

[Zas85] J.J. Zasio, "Non-Stuck Fault Testing of CMOS VLSI," *Proc. IEEE Compcon, Spring 1985*, pp. 388-391, 1985.

[Zor84] Y. Zorian and V.K. Agarwal, "Higher Certainty of Error Coverage by Output Data Modification," *Proc. ITC-84*, pp. 140-147, Oct. 1984.

[Zor86] Y. Zorian and V.K. Agarwal, "A General Scheme to Optimize Error Masking in Built-In Self-Testing," *Proc. FTCS-16*, pp. 410-416, Vienna, 1986.

[Zor87] Y. Zorian, *Optimized Error Coverage in Built-In Self-Test by Output Data Modification*, Ph.D. Dissertation, Dept. of Elect. Eng., McGill University, Montreal, PQ, Sept. 1987.