

**OPTIMIZED ERROR COVERAGE  
IN BUILT-IN SELF-TEST  
BY OUTPUT DATA MODIFICATION**

**Yervant Zorian**

**VLSI Design Laboratory  
Department of Electrical Engineering  
McGill University  
Montreal, CANADA**

**A thesis submitted to the Faculty of Graduate Studies  
and Research in partial fulfillment  
of the requirements for the degree of  
Doctor of Philosophy  
(Electrical Engineering)**

**September 1987**

**©1987 by Y. Zorian**

## Abstract

The concept of *Built-In Self-Test* (*BIST*) has recently become an increasingly attractive solution to the complex problem of testing *VLSI* chips. However, the realization of *BIST* faces some challenging problems of its own. One of these problems is to increase the quality of fault coverage of a *BIST* implementation, without incurring a large overhead. In particular, the loss of information in the output data compressor, which is typically a multi-input linear feedback shift register (*MISR*), is a major cause of concern.

In the recent past, several researchers have proposed different schemes to reduce this loss of information, while maintaining the need for a small area overhead.

In this dissertation, a new *BIST* scheme, based on modifying the output data before compression, is developed. This scheme, called *output data modification* (*ODM*), exploits the knowledge of the functionality of the circuit under test to provide a circuit-specific *BIST* structure. This structure is developed so that it can conveniently be implemented for any general circuit under consideration. But more importantly, a proof of effectiveness is provided to show that *ODM* will, on the average, be orders of magnitude better than all existing schemes in its capability to reduce the information loss, for a given amount of area overhead.

Moreover, the constructive nature of the proof will allow one to provide a simple trade-off between the reduction tolerated in information loss to the area overhead needed to affect this reduction.

## Résumé

La vérification des circuits intégrés à très grande échelle (VLSI) constitue un problème très complexe. L'approche des tests incorporés (BIST) est une approche récente qui est de plus en plus attrayante. Néanmoins, la réalisation de puces autotestables pose de sérieux problèmes aussi. Un de ces problèmes est d'améliorer la qualité de l'autovérification du circuit sans nécessiter une trop grande aire supplémentaire de silicium. La perte d'information résultant du vérificateur/compresseur de réponse (qui est le plus souvent implémenté sous forme de registre (Shift Register) à entrées multiples) constitue un des points d'intérêts majeurs de la recherche actuelle dans le domaine.

Récemment, plusieurs chercheurs ont proposés des approches visées à réduire la perte d'information résultant de la compression de la réponse du circuit. En général ces approches ne requiert que peu d'aire de silicium supplémentaire. La présente dissertation propose une nouvelle approche d'autovérification. Celle-ci est basée sur la modification de la réponse du circuit avant la compression. L'approche, appelée "output data modification" (ODM), utilise la description fonctionnelle du circuit afin de réaliser une structure de test propre à un circuit. Cette structure est développée afin qu'elle puisse aisément être implémenter pour n'importe quel circuit.

Le plus important est qu'une preuve d'efficacité est donnée. Celle-ci démontre qu'en général, pour une aire de silicium supplémentaire donnée, l'approche ODM résulte en une diminution de la perte d'information qui est de plusieurs ordres de grandeur inférieure aux autres approches précédemment proposées.

En outre, la nature constructive de la preuve permet de facilement établir le rapport entre la réduction de perte d'information désirée, et l'aire supplémentaire requise pour effectuer cette réduction.

## Acknowledgements

I wish to express my sincere gratitude to Professor Vinod K. Agarwal, who as supervisor provided invaluable and instructive guidance, as well as motivation and enthusiasm throughout the course of this research.

- I would also like to extend my thanks to Mr. André Ivanov for his grateful help in writing the french abstract, and making useful comments on the presentation of this material, and Mrs. Maida Kazanjian for her kind participation in preparing the graphics.

- Very special thanks go out to my wife and parents, without whose support and encouragement would have made the task considerably much more difficult.

Finally, I am grateful also to the National Sciences and Engineering Research Council of Canada and McGill University for providing financial support during the period of this research.



## Statement of Originality and Contribution to the Knowledge

The author of this thesis claims originality for the following contributions:

- The development of a *BIST* scheme, based on the output data modification concept, to optimize the error masking problem, which is one of the major concerns of *BIST*. This scheme provides the best implementation in the quality of *BIST* compared to all the earlier attempts. An analytical proof also developed in this thesis shows that for average cases, it provides tremendous reduction in error masking.
- The development of several designs for sequence generation. These designs are characterized by their capability to generate large numbers of distinct sequences (of length  $l$ ) using a small amount of hardware ( $O(\log(l))$  gates). These designs can also be adopted in various applications other than the *output data modification* scheme.
- The development of a heuristic algorithm which has the goal of determining a good approximation to a desired sequence that can be obtained for a given amount of hardware.
- The development of a trade-off concept in *BIST* designs to achieve optimal solutions between the two following constraints: the desired amount of test quality, and a tolerated amount of extra hardware. More specifically, this concept is elaborated for the case where test quality is measured by error masking probability and where the amount of extra hardware is measured in number of cubes.

## Table of Contents

<i>Contents:</i>	<i>Pages:</i>
Abstract . . . . .	i
Resume . . . . .	ii
Acknowledgements . . . . .	iii
Statement of Originality and Contribution . . . . .	iv
Table of Figures . . . . .	viii
Chapter 1: Introduction . . . . .	1
1.1 Background . . . . .	1
1.1.1 Cost of Testing . . . . .	2
1.1.2 Generating a Test Set . . . . .	2
1.1.3 Measures of Test Quality . . . . .	3
1.1.4 Applying Input Patterns and Analyzing Output data . . . . .	4
1.2 VLSI Testing Strategies . . . . .	5
1.2.1 Deterministic Testing Strategy . . . . .	5
1.2.1.1 Limitations of Deterministic Testing . . . . .	6
1.2.2 Universal BIST Strategy . . . . .	8
1.2.2.1 General BIST Model and Self-Test Operation . . . . .	9
1.2.2.2 Limitations of Universal BIST . . . . .	12
1.2.3 Circuit-Specific BIST Strategy . . . . .	13
1.3 Error Information Loss in BIST . . . . .	15
1.4 Dissertation Outline . . . . .	17
Chapter 2: BIST for Random Logic . . . . .	19
2.1 Preliminaries . . . . .	19
2.1.1 Linear Feedback Shift Registers . . . . .	21
2.1.2 Hardware Cost of BIST . . . . .	26
2.2 Input Pattern Generation Process . . . . .	27
2.2.1 Exhaustive Test-Based Techniques . . . . .	28
2.2.2 Pseudo-Random Test-Based Technique . . . . .	33

2.2.2.1 Establishing Pseudo-Random Test Length	34
2.2.2.2 Random Pattern Resistant Faults	36
2.2 Output Data Compression Process	37
2.3.1 Polynomial Division-Based Techniques	40
2.3.2 Parity Check-Based Techniques	44
2.3.3 Count-Based Techniques	46
Chapter 3: The Error Masking Problem and Output Data Modification	51
3.1 Introduction	51
3.2 The Error Masking Problem	52
3.3 Improving Error Coverage	57
3.3.1 Extended Compressor Approach	57
3.3.2 Segmented Test Set Approach	58
3.3.3 Multiple Level Compression Approach	60
3.3.4 Multiple Test Sets Approach	61
3.3.5 Multiple Compressions Approach	62
3.3.6 Variable Shift Compression Approach	64
3.3.7 Combined Compression Approach	64
3.4 The <i>ODM</i> Concept	67
3.5 The Proposed <i>ODM</i> Model	71
Chapter 4: Modifier Sequence Generation Methods	74
4.1 Introduction	74
4.2 Methods to Generate a Sequence by Segments	75
4.2.1 <i>EIS</i> Methods for Modifier Sequence Generation	76
4.2.1.1 <i>EIS</i> with Single Input and a Single Seed	76
4.2.1.2 <i>EIS</i> with Single Input and Multi-Seed Possibility	82
4.2.1.3 <i>EIS</i> with Double Inputs	84
4.2.1.4 Further <i>EIS</i> Possibilities	87
4.2.2 <i>ESS</i> Method for Modifier Sequence Generation	89
4.3 A Method to Generate Modifier Sequences by Totality	91
Chapter 5: Reduction in Deception Volume for Average Sequences	95
5.1 Preliminary Calculations	95
5.2 Essential Cubes and Isolated Vertices	98
5.3 The Number of Sequences with Isolated Vertices in a <i>k</i> -cube	99

5.4 Average Number of $k$ -cubes with Exact Number of Isolated Vertices	102
5.5 Weight of a $k$ -cube with $v$ Isolated Vertices	103
5.6 The Available Cubes and the Reduction in Deception Volume	104
Chapter 6: Realization of the ODM Scheme	108
6.1 The General BIST Design of the ODM scheme	108
6.1.1 The Standard Self-Test Facilities	109
6.1.1.1 Input Pattern Generator	109
6.1.1.2 P/S Compressor	111
6.1.2 The Modifier Block	114
6.1.3 The Improved Compressor	115
6.2 Circuit-Specific Implementation	118
6.2.1 A Procedure to Implement the ODM Scheme	119
6.2.2 Selection of a Limited Number of Cubes	123
6.3 Conclusive Remarks	129
Chapter 7: Results and Conclusions	131
7.1 Introduction	131
7.2 Simulation Results	132
7.3 Conclusive Remarks on the ODM Scheme	135
7.4 BIST Schemes Adopting the ODM Approach	138
7.5 Further Possibilities under ODM	139
References	142

## Table of Figures

<i>Figures:</i>	<i>Pages:</i>
Figure 1.1 A General <i>BIST</i> Model . . . . .	10
Figure 2.1 The General Form of an <i>LFSR</i> . . . . .	22
Figure 2.2 An Illustration of a Polynomial Division . . . . .	23
Figure 2.3 An Autonomous <i>LFSR</i> with Maximum Cycle Length . . . . .	25
Figure 2.4 Pseudo-Exhaustive Input Pattern Generation . . . . .	32
Figure 2.5 An <i>LFSR</i> as an <i>ODC</i> . . . . .	41
Figure 2.6 Multi-Input Shift Register ( <i>MISR</i> ) as <i>ODC</i> . . . . .	42
Figure 2.7 Parity Checker for an Output Data Stream . . . . .	45
Figure 2.8 Count-Based Compression Techniques for Single Output <i>CUTs</i> . . . . .	47
Figure 2.9 One's Counting for Multi-Output <i>CUTs</i> . . . . .	49
Figure 3.1 A Representation of an Output Data Compression . . . . .	54
Figure 3.2 Extended Compressor Approach . . . . .	58
Figure 3.3 Segmented Test Set Approach . . . . .	59
Figure 3.4 Multiple Level Compression Approach . . . . .	60
Figure 3.5 Multiple Test Sets Approach . . . . .	62
Figure 3.6 Multiple Compressions Approach . . . . .	63
Figure 3.7 Combined Compression Approach . . . . .	65
Figure 3.8 Uniform Deception Volume ( <i>MISR</i> ) . . . . .	68
Figure 3.9 Non-Uniform Deception Volume (One's Counting) . . . . .	69
Figure 3.10 The Modification Operation . . . . .	70
Figure 3.11 A <i>BIST</i> Model for the <i>ODM</i> Scheme . . . . .	72
Figure 4.1 Single Input Method . . . . .	83
Figure 4.2 Double Input Method . . . . .	85
Figure 4.3 Two Level Modification Technique . . . . .	89
Figure 4.4 Equal Size Segments Method . . . . .	90
Figure 5.1 Isolated Vertices in an Essential Cube . . . . .	99
Figure 5.2 The States of Three Isolated Vertices in an Essential Cube . . . . .	102
Figure 5.3 The Trade-Off Curves . . . . .	106

Figure 6.1	An <i>IPG</i> Design with Two <i>LFSRs</i> . . . . .	110
Figure 6.2	An <i>IPG</i> Based on Pseudo-Exhaustive Technique . . . . .	111
Figure 6.3	The <i>Improved Compressor</i> . . . . .	117
Figure 6.4	The Hardware Realization of the <i>Modifier Block</i> . . . . .	122

## Chapter 1

### Introduction

---

#### 1.1 Background:

The integrated circuit (IC) technology is continually moving towards higher scale and density factors, hence making possible the fabrication of circuits with a very large number of devices on a single chip. However, increases in circuit complexity, due to these technological improvements, have made testing such circuits a major problem [Williams 83].

Testing is performed in order to determine whether a circuit is operating correctly, or not, i.e. to find out if it is producing identical results to what it is originally designed for.

A testing procedure, in general, consists of three steps: generating a set of input patterns (i.e. test set); applying these to the circuit under consideration; and analyzing

the resulting output data. This procedure is usually performed at various levels during the production of a system [Mann 80] [Myers 83]. For instance, the dies are tested during fabrication, the packaged chips before insertion into the boards, the boards after assembly, and finally, the entire system is tested when complete. The complexity of the testing procedure at each level is determined by factors such as the time available for testing, the degree of access to internal circuitry, and the percentage of faults which are required to be detected.

#### 1.1.1 Cost of Testing:

As digital systems become more complex, their cost of testing becomes a major part of the cost of a system. Thus, there is much activity in trying to reduce the cost of testing by developing different testing strategies. It is, however, important to mention that the cost of finding a failing component depends, in addition to the testing strategy upon the level of testing (i.e. chip level, board level, ...) as well. In general, since detecting and isolating a fault at the board or higher packaging levels costs more than at the chip level [Williams 83]. To minimize the cost of testing, it is imperative that faults be detected as early as possible. Most of our discussions, throughout this dissertation will be concentrated on chip level testing, although the scheme proposed is applicable to other levels besides chip level testing.

#### 1.1.2 Generating a test set:

The testing procedure, as indicated previously, starts by generating a test set. Such a set is utilized in order to identify the circuits that have become faulty due to physical failures. The large number of possible failures dictates that a practical strategy to generate a test set should avoid working directly with these physical failures. Since, at



the chip level, one is not usually concerned with determining the exact physical failure, what is desired is merely to find out the existence or absence of any failure. Thus, in general, the effects of physical failures are described by a fault model. If a fault model accurately describes all the physical failures of interest, then, one only needs to generate a test set which detects all the faults in the fault model. The most widely-used fault model is that of a single line being permanently stuck-at a logic value of 0 (s-a-0) or 1 (s-a-1).

The presence of a given fault is said to be detected when an appropriate input pattern, applied to the circuit-under-test (CUT), causes an incorrect logic result (that is, error) at one or more output lines of the CUT. Different physical failures/faults may cause the same error under a given pattern. Given a fault model, a test set is considered

complete if it detects the entire set of faults in a CUT. Hence, it is essential that for every fault, there must exist at least one pattern by which it will be detected. It will be seen later, that despite its necessity a complete test set is often not easy to obtain for large circuits.

Various algorithms [Goel 81] [Fujjwara 83] have been developed to generate test sets for chip level testing. Conventionally, a test set at this level consists of either deterministic, exhaustive, or randomly generated patterns, depending on the testing strategy adopted. A discussion concerning these strategies will appear later in section 1.2.

### 1.1.3 Measures of Test Quality:

When considering a set of input patterns to test a complex circuit, one should first consider how good it is for detecting the set of faults in the circuit. Usually, the quality

of a test set is measured by the ratio of the faults detected to the total number of faults in the model. This ratio is termed as fault coverage.

Fault coverage is usually determined by a process called fault simulation. This process consists of simulating the application of every pattern in the test set to the set of faulty circuits (corresponding to the circuit to which a fault is injected) and comparing the responses to that of a fault-free circuit. However, such a simulation of the entire set of faults, particularly in large circuits with many tens of thousands of gates may take a prohibitive amount of computing time. As an alternative to simulation, several probabilistic measures have been proposed to estimate the fault coverage [Savir 83a] [Seth 85].

#### 1.1.4 Applying Input Patterns and Analyzing Output Data:

Once a set of input patterns is generated, it is then applied to the circuit under consideration. Depending on the testing strategy adopted, the source of input patterns is in some cases inside the chip itself, while in others in an external device. This will be discussed more fully in a later section. Similarly, the output data analysis is either performed internally to the chip or via an external device. It needs to be mentioned, that external devices, which serve the functions of generating input patterns and analyzing output responses, are often very expensive and very complex systems.

Whether performed internally or externally, the analysis of output data always consists of comparing the entire output response, or some function of it, with reference values representing the correct responses. Based on this comparison, if the two values differ, the chip is declared faulty or else fault-free.

The following section describes in brief some of the testing strategies that are best suited for complex VLSI chips.

## 1.2 VLSI Testing Strategies:

Different strategies to VLSI testing have been developed over the years [Williams 83] [Williams 84] [Wang 85]. The purpose of this section is to provide a short description of these strategies, outline their uses, and describe some of their limitations. The first strategy considered is the deterministic testing, which is probably still most common, although it has numerous limitations. One most significant limitation being that, it is heavily dependent on the structural information of the CUT. The second strategy, to be addressed in this section, is universal testing (i.e. CUT's structure-independent), which adopts the concept of building test facilities into a chip. Since the inclusion of test facilities is conventionally known as "built-in self-test" (BIST) [McCluskey 85a], the above mentioned strategy will be termed as the universal BIST. The third strategy, considered in this section, includes a new trend of BIST schemes [Schnurmann 75] [Barzilai 81] [Agarwal 83] [Zorian 84]. These schemes distinguish themselves from the previous schemes by having their self-test facilities adjusted (i.e. programmed) specifically for given circuits. We will classify these under a strategy termed as circuit-specific BIST strategy.

### 1.2.1 Deterministic Testing Strategy:

As mentioned in the previous section, the first step in the testing procedure consists of generating input patterns for a given CUT. Under the deterministic testing strategy, this step is based on the topological structure of the circuit, and its set of faults. To

derive an input pattern for a particular fault, a specific structure-dependent computation is needed. The input patterns resulting from these compose a deterministic test set. Such computations are usually performed by one of numerous algorithms known as automatic test pattern generation algorithms [Roth 66], [Goel 81], [Fujiwara 83]. To evaluate the quality of the test set, a fault simulation process is also required [Chang 70] [Armstrong 72] [Bose 82] [Hayes 82] [Levendel 81].

Under the deterministic testing, external test equipments are used to apply generated test sets, as well as to perform the output data analysis [Williams 83].

#### 1.2.1.1 Limitations of Deterministic Testing:

The continuous growth in the complexity of chips creates major problems in all three steps of the deterministic testing procedure. Due to these problems, the input pattern generation and application, as well as the output data analysis, have become extremely expensive steps. In order to better understand the magnitude of these problems, the difficulties of each of these three steps is briefly discussed in the following.

The test pattern generation problem under this strategy, is well recognized to be an NP-complete problem [Fujiwara 82] [Ibarra 75] even for combinational circuits. This implies an exponential growth in test generation time with the growth in hardware complexity. Various algorithms [Roth 66] [Goel 81] [Fujiwara 83] have been developed to reach more tractable growth rates for average circuits. An illustration in [Goel 80] shows that the test generation with such an algorithm, in general, consumes CPU time at a rate approximating  $G^2$  (where  $G$  is the number of gates in the circuit). While this growth rate is much more tractable than an exponential one, it still is not very attractive for large area chips. [Goel 80] projected upwards of 1000 hours of CPU time

to generate tests for a 100,000 gate structure, even assuming an algorithm which could determine tests without backtracking. Obviously, the costs of test pattern generation for complex VLSI chips could quite easily become intolerable.

Another major problem with complex VLSI chips is in performing a complete fault simulation (needed to determine the fault coverage) for a deterministic test set. It is observed that the computer run time is proportional to  $G^2$  to do fault simulation [Williams 79], and is proportional to  $G^3$  for both test generation and fault simulation [Goel 80]. Thus, small increases in gate count will yield to quick increases in run times. Several techniques have been reported to reduce the complexity of fault simulation [Parker 79] [Ulrich 74]. However, it still is a very time consuming, and hence expensive task [Williams 79], particularly for complex chips.

Furthermore, the two other steps in the testing procedure (applying input patterns and analyzing output data), as indicated above, face severe problems with large VLSI chips as well. These problems are in terms of time and volume. More specifically, the amount of time needed to apply a set of deterministic patterns, and to perform the output data analysis, has grown to the extent that it often results in an unacceptable test duration [McCluskey 85a]. In regards to the volume, or number of input patterns and output reference values, it has become too large to be handled efficiently by the test equipment hardware [McCluskey 85a].

Finally, another characteristic of the deterministic testing strategy mentioned earlier, is the dependence of test pattern generation and fault simulation on the structural information of a CUT. This is a limitation simply because a detailed structural (i.e. topological) information of a complex VLSI chip is often hard to obtain and to analyze.

### 1.2.2 Universal BIST Strategy:

The limitations of the deterministic testing strategy have led the attention of IC manufacturing and design communities to new methods for testing complex VLSI chips. One such method, which increases the controllability and observability of the circuits, and avoids the inadequacies of deterministic test generation and fault simulation for sequential circuits, is based on the idea of designing circuits for testability. This is realized by adopting some simple built-in testing techniques, known as design for testability techniques [Williams 83] [Mangir 83] [Muehldorf 81]. Conventionally, such techniques are in some cases in the form of general guidelines to be followed. In others, they are design rules to be implemented. Associated with these techniques is their cost of implementation and return of investment [Akers 77] [Eichelberger 78] [Hayes 74] [Funatsu 75].

It is important to mention that most of the design for testability techniques came as a response to the difficulty of deterministic test generation for sequential circuits. Hence, in order to avoid this difficulty, a number of companies including IBM [Eichelberger 77], Sperry Univac [Stewart 77], Nippon Electric [Funatsu 75] and Fujitsu [Ando 80] have developed special techniques commonly known as scan techniques. These techniques suggest a structured design approach in which all sequential elements are organized so that their contents can easily be controlled and observed. Using a scan technique reduces the test generation problem for a sequential circuit to the test generation problem for a combinational circuit alone [Williams 83] [McCluskey 84b]. Usually sequential elements (such as latches and flip-flops) tested under these techniques, are first switched from their normal mode of operation to a test mode. During the test mode, these elements become threaded together into one or more shift registers. This makes it possible to first scan in arbitrary test patterns through them, and then scan these patterns out to compare

the output data with the correct response. These techniques are becoming commonly accepted methods and therefore increasingly implemented by the manufacturers[].

As a result of adopting a scan-based technique, the sequential testing problem is reduced to one for combinational circuitry only. Hence, the main task remains to develop a testing strategy which solves the problems of deterministic testing for combinational circuits. In other words, the major intent is to provide a testing strategy that eliminates the problems of test generation and fault simulation for combinational circuits, avoids the expensive test equipments, and is also independent of the CUT's structural details.

A new testing strategy promises to realize all the above mentioned intentions by providing a structure-independent and on-chip testing strategy. It is termed as universal BIST strategy. This testing strategy moves some or all of the test equipment functions onto the chip itself, or onto the board on which the chips are mounted. Therefore, it is called a built-in self-test strategy. Furthermore, it is called universal because the structure of its self-test facilities holds a general design, independent of the CUT's structure (i.e. the facilities are not programmed specifically for a given circuit).

Under the universal BIST, various built-in self-test schemes for random combinational logic [McCluskey 85a], PLAs [Treuer 85] and memories [Sun 84] have been developed. Such schemes are useful tools in the testing of complex VLSI chips. More details about these BIST schemes and in particular about BIST techniques used for random combinational logic are described in the first two chapters of this dissertation.

#### 1.2.2.1 General BIST Model and Self-Test Operation:

Due to the increasing use of scan-based techniques, the main emphasis of current BIST schemes is to provide close to a 100% testing of the combinational circuitry.

The implementation of a *BIST* scheme implies the addition of self-test facilities, that enable the resulting structure (i.e. the *BISTed* design) to apply the input patterns and analyze the output data. Figure(1.1) shows a general *BIST* model which consists of the combinational *CUT* itself and the extra blocks for self-testing.

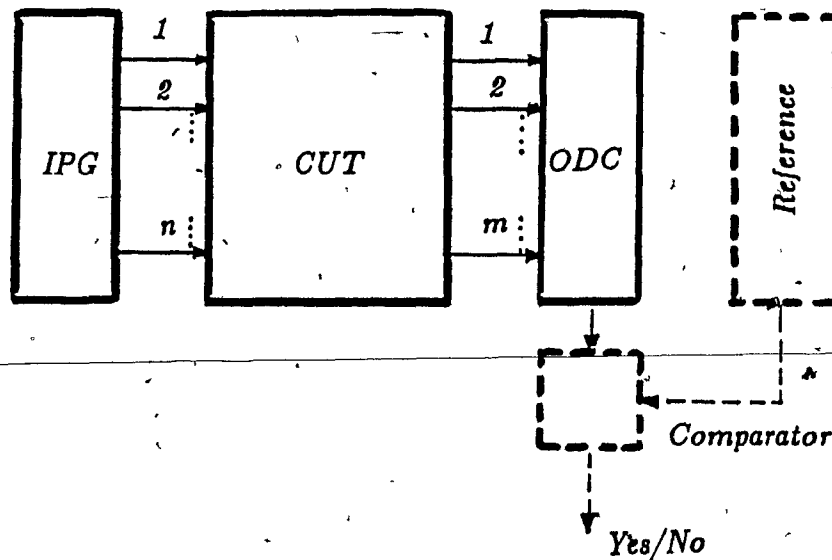


Figure (1.1) A General *BIST* Model

In a typical *BIST* model, the extra blocks remain transparent to the user during the normal mode of operation as long as the original circuit functions according to its specifications. During the self-test mode, one of the self-test blocks called *IPG* (input pattern generator) generates the input patterns internally and applies them to the circuit in question. Conventionally, the test sets used in *BIST* schemes are either exhaustive, i.e. consisting of all possible input patterns, or pseudo-random, i.e. consisting of pseudo-randomly generated patterns. The test patterns for both cases (i.e. exhaustive, or pseudo-random) can be generated internally, as will be discussed in the section 2.2, by simple *IPGs* during the self-test mode. Hence, there is no need for the difficult test pattern generation processes that are needed under the deterministic testing strategy. Such an internal pattern generation possibility solves also the difficult requirement of



storing the test patterns in advance, and thus avoids a large part of the memory required to store these patterns.

Furthermore, most of the BIST schemes reduce the amount of output data prior to the final step of output data verification. This process of lessening the amount output data is well known as output data compression, and performed by a block termed output data compressor (ODC). This block collects the actual output data of a CUT and compresses it into a signature of the output response. The intent behind this lessening is to reduce the amount of memory required to store the voluminous reference values needed to verify the actual output data of a CUT.

Finally, in order to complete the output data analysis, the last two blocks perform the verification step. Wherein, the observed signature in the ODC is compared by the Comparator with the fault-free value stored in the reference block. Hence, if the two signatures differ, it is safe to predict that the CUT is faulty. And if the two are identical, then the CUT is conventionally declared fault-free. However, identical signatures do not guarantee a fault-free circuit, since a faulty circuit can produce some output combination which gets compressed into the same signature as the fault-free one.

Under various BIST schemes, the self-test blocks in Figure(1.1) are realized by different devices (i.e. circuit structures). It is important to note that these devices are intended to be simple and nonexpensive, so that they require tolerable amounts of hardware. The most commonly utilized IPG consists of an extremely simple device, called linear feedback shift register (LFSR) which has the capability of generating both exhaustive and pseudo-random patterns [Peterson 75]. The output data compression itself is often performed by simple devices as well, e.g. a counter for One's Counting [Savir 80] and Transition Counting [Hayes 76b] techniques, or a multi-input linear feedback shift register (MISR) for another compression technique known as polynomial division

[Benowitz 75].

The characteristics of input pattern generation and output data compression techniques, as well as their specific devices, are presented in more detail in chapter 2. It can be noted that the use of pseudo-random pattern generation followed by compression of output data by polynomial division is being increasingly adopted by many reported BIST schemes [Konemann 79] [Bhavsar 81] [Bardell 82] [Komonytsky 82]. Hence, this use has become an almost standard form of testing in universal BIST. Furthermore, this standard BIST has recently been adopted by several manufacturers in their products [Danniels 81].

#### 1.2.2.2 Limitations of Universal BIST:

The current universal BIST schemes have different limitations. One of these is that if schemes with exhaustive testing are used for complex VLSI chips, then a circuit with a large number of inputs may require an exceedingly large test set, with its consequent unacceptable test time [Bözorgui-Nesbat 80].

On the other hand, if pseudo-random testing is used, certain faults, known as random pattern resistant faults, may exhibit extreme reluctance to detection by the pseudo-random patterns [Eichelberger 83]. Therefore, such faults may result in unsatisfiable fault coverages.

Yet another drawback with universal BIST is faced if a faulty circuit is declared as fault-free. As mentioned earlier, this may happen when an erroneous output data is compressed by some ODC into a signature identical to the fault-free one. It is obvious that this drawback also affects the quality of test, since it reduces the expected fault coverage of a test set [Smith 80] [Bhavsar 84]. In particular, this phenomenon which

occurs under current *BIST* schemes for random logic using output data compression will be the focus of interest throughout this dissertation.

### 1.2.3 Circuit-Specific *BIST* Strategy:

Although an ideal *BIST* structure should be a completely general one, in the sense that it should be applicable to any *CUT*, in order to avoid some of the limitations in universal *BIST*, a new strategy which considers circuit-specific self-test structures is under development [Agarwal 81] [Agarwal 83] [Zorian 84] [Tang 84] [Chin 84]. This new strategy, termed here as circuit-specific *BIST* strategy, is mainly aimed at optimizing the test quality, while maintaining most of the advantages of universal *BIST*.

With the above mentioned strategy, a typical *BIST* scheme utilizes certain information about the *CUT* in question to program the self-test facilities. This information is generally the functionality (i.e. the functional information) of a *CUT*. Conventionally, despite that such self-test facilities have a general design, they need to be programmed specifically for given *CUTs*. Notice that the three limitations of universal *BIST* mentioned in section 1.2.2.2, can be avoided or reduced by adopting such circuit-specific *BIST* schemes. The overcoming of these limitations by circuit-specific *BIST* schemes will only be briefly discussed next. More thorough discussions appear in the subsequent chapter.

The first case considers the test length problem for complex *VLSI* chips under exhaustive testing. Various *BIST* schemes, known as pseudo-exhaustive testing schemes, appeared recently in the literature [Wang 84] [Tang 84]. They suggest different techniques to reduce this test length problem by adopting circuit-specific *IPGs*. Such a technique typically exploits the functional information of a *CUT* to determine the set

of inputs which drive each output, and then accordingly set a specific *IPG* that will provide a reduced test length. Several pseudo-exhaustive techniques are described in section 2.2.1.

The second case to illustrate the use of circuit-specific *BIST* deals with the attempts aimed at detecting random pattern resistant faults, in order to increase the fault coverage of schemes based on pseudo-random test. Some of the attempts, to solve the problem of random pattern resistant faults, suggest ways to modify a regular pseudo-random *IPG* for a given *CUT* (see section 2.2.2.2). One modification is to generate unequiprobable (i.e. biased) random patterns. These have been shown to provide higher possibility of detecting the resistant faults [Schnurmann 75] [Chin 84] [Wundelich 87].

Another question of major concern under most of the current *BIST* schemes for random combinational logic is the problem of declaring a faulty circuit as fault-free due to output data compression [McCluskey 85a] [Williams 84]. This problem can also be reduced by adopting circuit-specific *BIST* schemes [Agarwal 83] [Zorian 84] [Li 87]. This reduction is realized by exploiting the functionality of the *CUT*, and accordingly providing circuit-specific *IPGs* and *ODCs*. The main issue of the remainder of this dissertation is to introduce, implement and analyze a complete *BIST* scheme which has the ability of drastically reducing the problem of declaring a faulty circuit as fault-free by modifying the output data [Zorian 84] [Zorian 86c].

It is important to indicate, at this stage, that the circuit-specific *BIST* strategy in general might be considered as a promising strategy since it tends to maintain the advantages, and avoid the problems of the two previous testing strategies (deterministic and universal *BIST*). To be more specific, considering implementation costs, the

7

0

circuits-specific BIST strategy avoids the large expenses of input pattern generation, fault simulation and test equipments under the deterministic testing strategy, while it tends to maintain the ease of implementing BIST facilities (IPGs and ODCs). In other respects, it avoids the dependence on the structural details of a given CUT needed under the deterministic testing. However, it usually utilizes the functionality of the CUT in question. Finally, such a strategy can improve the test quality of Universal BIST by minimizing the problems of error information loss and random pattern resistant faults.

### 1.2 Error Information Loss in BIST:

As discussed earlier, the compression of a CUT's output response leads, in general, to a loss of error information. Such a loss results in a reduction in the test quality of the overall BIST design, since the number of faults which remain undetected because of this loss, causes a reduction in the expected fault coverage [McCluskey 85a] [Bhavsar 84]. It has to be mentioned that faults in the BIST hardware itself may also lead to information loss. Actually, various BIST schemes including those using standard BIST might reach considerably low fault coverages due to error information loss. Such results are shown for example in [Saxena 85] [Zorian 86b]. Numerous BIST schemes intended to reduce the loss of error information have been reported in the literature [Hassan 83] [Hassan 84a] [Bhavsar 84] [Hlawiczka 86]. However, for the amount of hardware overhead that these schemes require, the reduction rate provided is limited (analysis found in section 3.3). Consequently, there is no provision to obtain better reduction rates under these schemes [Bhavsar 84].

The objective of this dissertation is a new BIST scheme which not only provides the best implementation in the quality of BIST over all the earlier attempts, but also tends to solve the problem of error information loss by drastically reducing its rate

to insignificant levels. Hence, this scheme provides a very high insurance that if an output data is ever erroneous, then its corresponding signature will differ from the fault-free signature. Such an insurance is particularly suited for very high test quality requirements ( 1 failure in a  $10^5$  or  $10^6$  parts etc... ). Moreover, the hardware overhead required to implement this scheme is not more than that used in the previous attempts.

Importantly, the amount of loss in error information for a BISTed design depends upon two factors. The first factor is the function adopted for output data compression, whereas the second consists of the CUT's error-free output data itself. None of the previous attempts takes both factors into consideration (see section 3.3). In effect, they suggest improving the effectiveness of the standard form of BIST by optimizing only one of the factors: the output data compression function [Bhavsar 84] [Carter 82a]. The BIST scheme presented in this dissertation utilizes the dependence on both factors. The above mentioned superiority of our scheme is actually the result of properly optimizing both of these factors, and consequently achieving tremendous increases in the effectiveness of standard BIST. The optimizations of both factors, under our scheme, as will be shown in section 3.4, are not independent of each other. In effect, an appropriate function for output data compression will first be determined. Then, accordingly, the output response data will be optimized. This optimization (of the second factor) can be realized by modifying the output data of a fault-free CUT into a modified output data which results in a substantial reduction in error information loss [Agarwal 83]. Moreover, this modification must be done by an easily implementable method. The BIST scheme developed throughout this dissertation utilizes such a modification to optimize its output data [Zorian 86c] by adopting an easily implementable method [Zorian 84] [Zorian 86a]. Hence, this scheme is termed in the remainder of this dissertation as output data modification (ODM) scheme.

The ODM scheme, as will be shown, consists of an adjustable, i.e. programmable, BIST structure (i.e. IPG, ODC, etc...), in order to perform modification of a given output data. Hence, this structure is specific to the CUT in question. Thus, accordingly the ODM scheme is classified under the circuit-specific BIST strategy.

In other respects, the BIST structure itself will be shown to have the ability of being very conveniently implemented for any general circuit. Furthermore, it is analytically proved (in chapter 5) that for any average case, the ODM scheme provides tremendous improvement in test coverage. Moreover, the constructive nature of this proof provides a whole range of trade-offs between the improvement in the desired test quality and the overhead in silicon area needed to affect this improvement.

#### 1.4. Dissertation Outline:

The remainder of this dissertation is organized as follows:

In chapter 2, a review of basic BIST techniques for random logic, used under universal and circuit-specific BIST strategies, is presented. These basic techniques are used for input pattern generation and output data compression processes. Most of these techniques will also be used for the ODM scheme.

In chapter 3, the error masking problem is discussed. An analysis of the existing BIST schemes aimed at reducing this problem is provided. Finally, the output data modification concept is introduced, and its BIST model demonstrated.

In chapter 4, various methods are developed to generate a specific element of the ODM scheme: the modifier sequence. This sequence, which is based on the functionality of a CUT, is shown to be easily generated by non-expensive circuitry.

In chapter 5, an effectiveness proof of the ODM scheme for average cases is developed. A useful trade-off between the improvement in desired test quality and the area overhead needed to affect this improvement is discussed as well.

In chapter 6, the realization of the ODM concept as a BIST model is discussed. Then, given a CUT, the implementation of the scheme is demonstrated, and in particular, the circuit-specific programming of its Modifier block is detailed.

Finally, chapter 7 concludes the dissertation with some simulation results and concluding remarks. There it will be shown that the ODM scheme can indeed serve a very useful role in providing high fault coverage through BIST for a minimal increase in the area overhead.



## Chapter 2

### BIST for Random Logic

#### 2.1 Preliminaries

The concept of *BIST* has in recent years become an invaluable tool in the testing of complex *VLSI* chips, especially with the outgrowth of scan-based techniques. This is evidenced by the several commercial chips with *BIST* implementations [Kuban 83] [Gelsinger 86]. Particularly, the current growing usage of application-specific integrated circuits (*ASIC*) has further increased the value of *BIST*, since the expensive procedure needed for the deterministic testing strategy cannot be justified for the limited production runs of *ASICs*. Several aspects concerning the importance of *BIST* and the benefits it provides were demonstrated in chapter 1, along with the differences between its universal and circuit-specific strategies.

One of the most useful aspect of BIST is its ability to test deeply embedded logic, PLAs and memories. It has now been established that BIST for regular structures such as PLAs and memories can quite efficiently provide a guaranteed fault coverage [Treuer 85] [Saluja 87] which does not require any fault simulation. On the other hand, BIST schemes for non-regular structures, such as random combinational logic, have not yet evolved to a guaranteed fault coverage environment. For most such schemes, the only ensured coverage is in terms of error coverage, as will be discussed in the following chapter. Various attempts [Hassan 84a] [Bhavsar 84] [Carter 82a] to increase this error coverage in order to increase the certainty of BIST for random combinational circuits have been reported. The ODM scheme, presented throughout this dissertation, has the same objective. However, it provides a tremendous improvement in error coverage at a very low additional cost. In the remainder of this dissertation, only built-in self-test for random combinational logic is treated.

In order to more easily understand the ODM design and its implementational aspects in the following chapters, it is important to illustrate some basic BIST techniques in advance. A review of such techniques and some analysis of their limitations are presented in this chapter. More specifically, the BIST techniques addressed here cover exhaustive and pseudo-random input pattern generation, as well as various output data compression techniques like polynomial division, parity check and count-based techniques. Most of these BIST techniques are utilized by different existing BIST schemes as well as to implement the ODM scheme.

Before reviewing and analyzing the above mentioned BIST techniques, it is helpful to direct one's attention to how these techniques are used by the ODM scheme. A typical input pattern generator under the ODM scheme has to provide two sets of patterns simultaneously: one pseudo-random and the other exhaustive. Thus, it is necessary to

address both pattern generation techniques in this chapter. In regard to the output data compression process under ODM, it is performed by two succeeding steps. The first step uses a polynomial division-based compression technique (i.e. MISR); whereas the second step utilizes a count-based technique. Hence, most of the BIST techniques introduced in this chapter will be useful in the following chapters, where the realization of the ODM scheme is presented. Prior to the illustration of these techniques, a well-known self-test device, called LFSR, is described.

### 2.1.1 Linear Feedback Shift Registers:

A BIST design, in general, consists of two principal units for self-test purposes (Figure(1.1)). These are an IPG which generates the input patterns and applies them to the CUT inputs, and an ODC which collects and then reduces the large amount of output data prior to its evaluation. These two units are often implemented by a device called a linear feedback shift register (LFSR). Conventionally, LFSRs are used to build pseudo-random generators [Golomb 82], polynomial division-based compressors [Benowitz 75], and store address generators [Hsiao 77] [Wang 82]. In order to well understand the capabilities and the limitations of an LFSR in a BIST structure, a brief description of its operation, structure and characteristics is provided in the following.

The mathematics, upon which an LFSR is based, is essentially the same as the basis for algebraic coding theory. A more detailed treatment of the material in this section can be found in [Peterson 75]. Coding theory treats binary sequences as polynomials with binary coefficients, where each bit in a sequence is the coefficient of a unique power of  $x$ . For example, the 5-bit sequence 11001 is represented by the 4-th degree polynomial  $m(x) = x^4 + x^3 + 1$ . Here, the degree of a polynomial is the largest power of  $x$  in a term of

the polynomial with a non-zero coefficient. The arithmetic, which must be used in the manipulation of these polynomials, is the arithmetic of the coefficients over the Galois Field of two elements  $GF(2)$  [Peterson 75]. Notice the result of dividing, multiplying or adding two binary polynomials also yields a polynomial in  $x$  with binary coefficients. For instance, the division of polynomial  $m(x)$  by  $p(x)$  yields a quotient polynomial  $q(x)$  and possibly, a remainder polynomial  $r(x)$ :

$$\frac{m(x)}{p(x)} = q(x) + \frac{r(x)}{p(x)} \quad (2.1)$$

**Example (2.1):** Suppose that  $m(x) = x^7 + x^3 + x$  is divided by  $p(x) = x^5 + x^3 + x + 1$ , then the quotient polynomial is  $q(x) = x^2 + 1$ , and the remainder is  $r(x) = x^3 + x^2 + 1$ .

LFSRs can be used to mechanize polynomial division. An LFSR is a linear sequential network [Elspas 59] composed of interconnections of two types of elements: memory elements (e.g. D-flipflops) and Exclusive-OR gates. Figure (2.1) represents the general form of a  $k$ -stage LFSR, where memory elements are represented by squares; and Exclusive-OR gates are represented by the symbol  $\oplus$ .

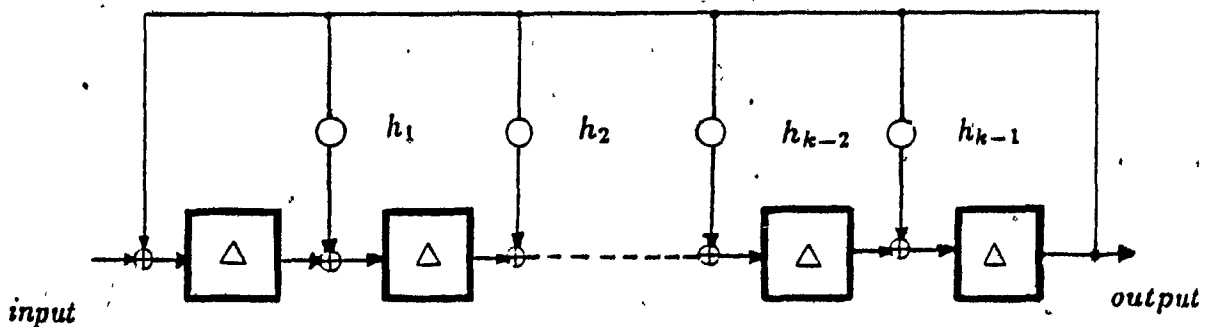


Figure (2.1) The General Form of an LFSR

The memory elements together form a shift register. The output of the last stage of this register is fed back to the inputs of some of these memory elements via Exclusive-OR gates. These gates give the shift register its linear property [Golomb 82]. The feedback connections which characterize an LFSR are represented by a polynomial known as the LFSR's characteristic polynomial  $p(x)$ :

$$p(x) = 1 + h_1x + h_2x^2 + \dots + h_{k-1}x^{k-1} + x^k \quad (2.2)$$

where  $h_i$  is equal to one if the corresponding feedback path  $h_i$  is a closed circuit; otherwise, it is equal to zero. The external input line shown in Figure(2.1) allows a serial feed of data.

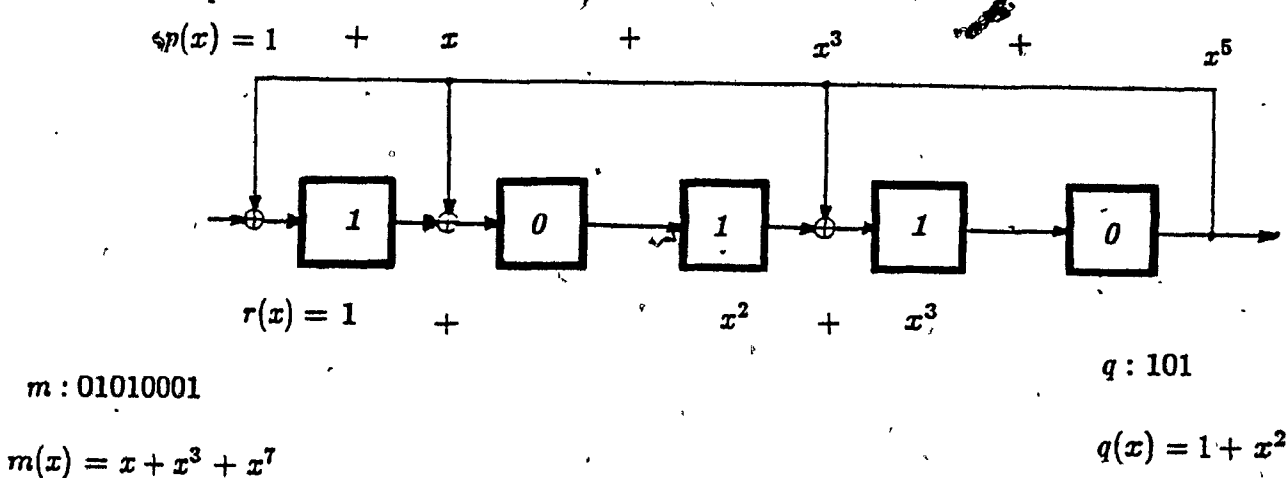


Figure (2.2) An Illustration of a Polynomial Division

The operation performed by an LFSR is equivalent to the process of dividing an input polynomial  $m(x)$  by the characteristic polynomial of the LFSR  $p(x)$ . Specifically, the polynomial division  $m(x)/p(x)$  (equation 2.1) is performed by initializing the LFSR to all 0's, and serially shifting into the LFSR via its input, the input sequence that  $m(x)$  represents. Due to this division, a new sequence is shifted out serially from the LFSR output (i.e. the last stage). After the last input bit has been shifted into the LFSR,

a content remains in the LFSR. The new sequence shifted out of the LFSR represents the quotient polynomial of the division,  $q(x)$ ; whereas the LFSR content represents the remainder polynomial  $r(x)$ , shown in equation (2.1). The polynomial division in Example(2.1) is realized by the LFSR shown in Figure(2.2).

Conventionally, when polynomial division is adopted to perform output data compression in a BIST design, an LFSR is utilized as the ODC. In such cases, the ODC's input sequence (i.e. input polynomial  $m(x)$ ) is the output sequence from the CUT. After the division process has been performed, the LFSR content (i.e. the remainder of the division  $r(x)$ ), represents the compressed form of the CUT's output sequence. Generally this remainder is termed as the signature (of the output response). Hence, the LFSR is sometimes called a signature analyzer [Frohwerk 77] [David 78].

Most of the BIST schemes adopt LFSRs for input pattern generation as well. This is possible when a test set of pseudo-random or exhaustive patterns is required. An LFSR that typically serves for input pattern generation is the autonomous LFSR, i.e. the one with no external input string applied to it. The sequence of test patterns generated from such an LFSR consists of its succeeding contents. Therefore, with each transition of state, the new LFSR content serves as the next pattern. Since an LFSR content appears on the outputs of its memory elements, the outputs of these elements can be directly connected to their corresponding CUT inputs. The only activating force of an autonomous LFSR is the feedback connections to the Exclusive-OR gates because there is no external input sequence applied to it (i.e. a constant 0 is applied to the external input). Consequently, given the characteristics of an autonomous LFSR and its initial state, the entire sequence of patterns can be determined in advance.

Each autonomous LFSR has a specific cycle, at the end of which the sequence of its patterns repeats. The cycle length (period) of an autonomous LFSR is determined by

its characteristic polynomial (in some cases, on the initial state as well). The maximum cycle length possible (upper bound) is  $2^k - 1$ . This will consist of all non-zero distinct patterns. This arises provided that the LFSR starts in a non-zero initial state, and that its characteristic polynomial belongs to a certain class [Gschwind 75]. This is the class of primitive polynomials, for which tables can be found in [Golomb 82] [Peterson 75], and many other publications. Figure(2.3) demonstrates an LFSR which generates such a set of patterns.

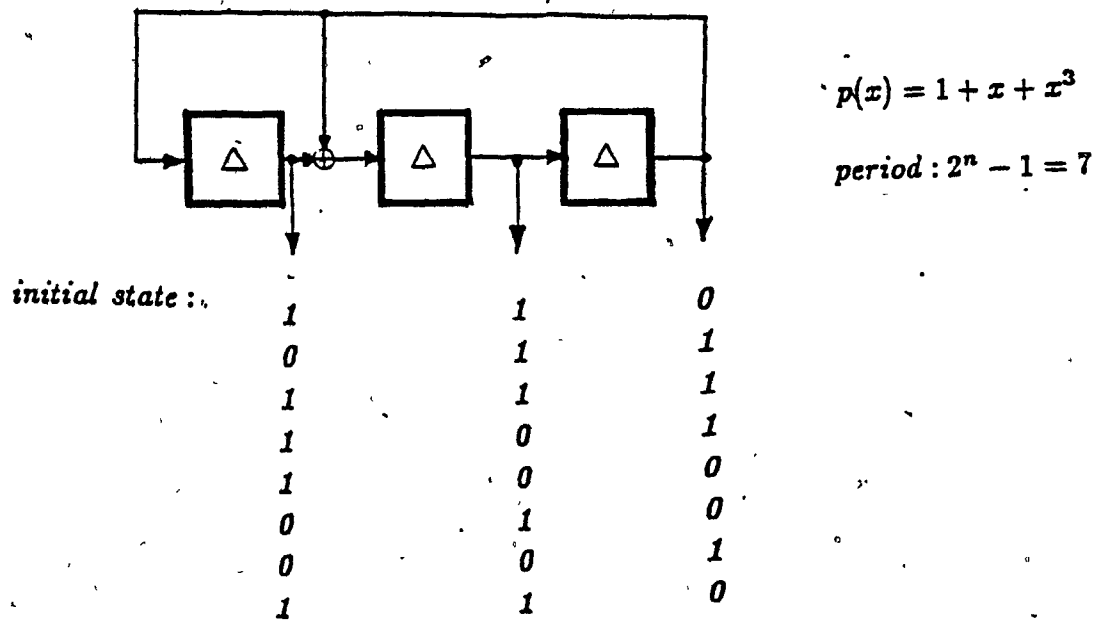


Figure (2.3) An Autonomous LFSR with Maximum Cycle Length

The above mentioned property of generating  $2^k - 1$  distinct patterns provides a useful application of LFSR's in BIST. This is the use of maximum cycle length LFSRs as IPGs for BIST schemes with exhaustive testing [Sedmak 79] [McCluskey 81].

Furthermore, the patterns generated by an LFSR with maximum cycle length possess another very useful property which can be observed in the order of the patterns' appearance [Muehldorf 81]. Any segment, out of the entire sequence of patterns gener-

ated holds properties similar to random patterns [Golomb 82]. It must be stated that the generation of these patterns cannot be considered truly random. One of the reasons being that in this case, no pattern is repeatable until all of the other patterns appear. Hence, these patterns are usually termed as pseudo-random patterns.[Golomb 82].

In BIST schemes, the ability of LFSRs to generate pseudo-random patterns is considered to be one of their most appealing properties. This is because it is established that applying pseudo-random patterns in sufficient numbers (but still much smaller than the set of all possible patterns) is often very effective for detecting faults in combinational [Agrawal 75] [McCluskey 85a] [Chin 87] as well as in sequential CUTs [Losq 78].

Moreover, another appealing property of LFSRs characterized by primitive polynomials (i.e. maximum cycle lengths) renders LFSRs useful for output data compression. It has been shown that for ODCs, primitive polynomials are preferable to non-primitive polynomials, because they guarantee the detection of all single-bit errors [Bhavsar 85]. Primitive polynomials are also preferred to non-primitive polynomials because they result in less error information loss. This is true for the case where specific error patterns are considered [Hassan 83], as well as for the more general case with different error occurrence probabilities [Williams 86] [Williams 87]. However, to predict the effect of the choice of a particular primitive polynomial on the fault coverage of a BIST scheme is still an open problem [Ivanov 87].

### 2.1.2 Hardware Cost of BIST:

Ever since the introduction of BIST techniques, the related hardware overhead, and cost of implementation, have been of primary concern [Williams 84]. Conventionally, self-test circuits needed to implement the BIST techniques are either added to



the original CUT, as in [Bhavsar 81], or formed by reconfiguring some of the existing elements in the CUT, like in [Konemann 79]. In both cases, some extra hardware cost arises. Obviously, such a cost depends on the design and implementation of the chosen BIST technique as well as the CUT itself. Various methods have been introduced to investigate the hardware cost of BIST designs [Ohletz 87]. From [Bardell 82] (where three different BIST schemes are described and compared for cost and performance), it appears that a 5% threshold is current practice. Higher levels may be acceptable if test cost is reduced enough. In the implementation of the ODM scheme as well, the hardware cost will be shown in chapter 6 to be of primary concern.

Various BIST schemes use different techniques to design the two principal self-test blocks. The remainder of this chapter treats a number of these techniques. This treatment will serve as a basis to understand the implementational aspects of the ODM scheme in the following chapters.

## 1.2 Input Pattern Generation Process:

The existing BIST schemes for combinational circuits suggest several techniques to perform input pattern generation. The pattern generation techniques of interest here, can be classified under the two most common types in BIST: exhaustive and pseudo-random. These two types are both of interest because they are needed to implement an IPG in the ODM scheme.

It is necessary for an IPG to produce a repeatable test set, and for the IPG to send distinct start and stop signals to the ODC. Furthermore, if the CUT and ODC memory elements, it must be possible to repeatably initialize them to a known state before the

start signal is issued. These forementioned requirements are to ensure that for a given circuit, the same signature results for the fault-free CUTs.

### 2.2.1 Exhaustive Test-Based Techniques:

Exhaustive testing implies the application of all possible  $2^n$  input patterns to a CUT, where  $n$  is the number of the input lines. The important advantage of this type of testing is that it is CUT-independent in the sense that no information about the CUT is required, except the number of its input lines.

Several schemes under universal BIST suggest the adoption of exhaustive testing to ensure complete fault coverage [Sedmak 79] [McCluskey 81] [McCluskey 82] [Agarwal 83]. Examples in [Savir 80] and [Barzilai 81] couple exhaustive testing with One's counting as the output data compression technique, while those in [Susskind 81] and [Muzio 82] couple exhaustive testing with the accumulation of Walsh coefficients.

The implementation of an IPG which performs exhaustive testing in a BIST design requires a device capable of generating all possible input patterns. An obvious candidate to perform this task is a binary counter. But since the order of the applied patterns is not important in an exhaustive test set, it is possible to use a maximum cycle length LFSR that generates all but one of the exhaustive patterns. Furthermore, the use of such an LFSR is also more efficient, since its hardware implementation is known to consume less area overhead than a binary counter of the same size. However, for the LFSR to generate the missing all-zero pattern, a modification of the maximum cycle length LFSR is needed. One possible modification is to add a scan path to scan-in the all-zero state through the LFSR [McCluskey 84b]. Another is to add extra gates to autonomously cycle through the all-zero state [McCluskey 86]. Yet another is to adopt

a systematic technique to design an *LFSR* which cycles through all the states [Wang 86a] [Wang 86c].

The major concern with exhaustive testing is that circuits with many input lines require a very large number of input patterns which implies a too long testing time. For instance, if a circuit has more than 25 input lines, a 100 nanosecond clock would take more than one second to complete the testing process. Due to this concern, different attempts have been made to reduce the number of input patterns when exhaustive testing becomes infeasible. To notice is that most of these attempts utilize some information about the circuit under consideration, and therefore, produce circuit-specific *BIST* structures.

In order to reduce the large number of input patterns required, some techniques suggest partitioning the circuit. This implies decomposing the *CUT* into manageable partitions [McCluskey 81]. Different partitioning techniques are described in [Bozorgui-Nesbat 80], [McCluskey 81] [DasGupta 84] and [Archambeau 84]

Still to reduce the test length and retain the advantages of exhaustive testing, several other *BIST* schemes suggest techniques which apply exhaustive patterns to portions of the *CUT* rather than to the entire *CUT*. This is realized by a back-tracing procedure whereby the actual number of inputs that drive every output is determined [Bottorff 77]. These input pattern generation techniques are known as pseudo-exhaustive techniques. More specifically, the concept underlying pseudo-exhaustive testing is the following. Since many combinational circuits may have outputs that depend on only a subset of the inputs, it is possible to exhaustively test the entire circuit by applying an exhaustive set of patterns to each subset of inputs, and then to observe the results on each corresponding output [McCluskey 85a]. In fact, to determine the subset of inputs upon which each output depends to design the required circuit-specific *IPG*, it suffices

to know the functionality of the CUT. It is important to mention that, the implementation of the pseudo-exhaustive techniques does not require any topological modifications to the CUT. The simplest IPG device that efficiently implements pseudo-exhaustive techniques is undoubtedly the LFSR. This is because some LFSRs have the capability of generating exhaustive test sets on some subsets of their outputs [Wang 84]. Such LFSRs require specific adjustments (hardware programming) according to the CUT. Several pseudo-exhaustive techniques are briefly described in the following.

The first pseudo-exhaustive technique appeared in [Barzilai 81]. It suggests the following. Given a CUT with  $n$  inputs, check if some of these inputs can share the same test signal. If  $n - n'$  inputs ( $n' < n$ ) can share the test signals with the other  $n'$  inputs, then the test length becomes  $2^{n'}$ . A binary counter of size  $n'$  is suggested to generate these input patterns. The problem with this technique is that when  $n'$  is close to  $n$ , the test length may still be too long.

To resolve this test length problem, another pseudo-exhaustive technique called verification testing was proposed [McCluskey 82], [Tang 83] [McCluskey 84]. Verification testing is based on the derivation of a constant-weight test set for a circuit without partitioning the circuit. The constant-weight test set is shown to be a minimum-length test set for most of the circuits. However, the major problem with this technique is that for circuits which require higher ( $m$ -out-of- $n$ ) codes (e.g. a 10-out-of-20 code), constant-weight counters become very costly to implement.

It may be more economical to use an LFSR instead of some form of counter for pseudo-exhaustive testing, even though this sacrifices the minimum test length property.

One LFSR-based technique suggests using a combination of LFSR's and shift registers [Barzilai 83] [Tang 84] for input pattern generation. It uses linear codes [Peterson 75] to find the desired generator polynomial for the LFSR. This technique is most useful

when the maximum number of inputs upon which an output depends is much less than  $n$ . However, it is difficult to find the suitable generator polynomial. Moreover, the technique usually requires at least two initial states which may have to be stored in a ROM. This storage can become a burden when there are many such circuits to be tested.

The forementioned storage problem can be substantially reduced by using condensed LFSR's for input pattern generation [Wang 84]. The circuit-specific designs based on this technique are very simple to implement. A designer can construct the condensed LFSR by simply using the formula given in [Wang 84] and choosing the primitive polynomial from a reference text such as [Peterson 75]. More specifically, according to this technique, an LFSR of size  $n$  with a characteristic polynomial  $p(x) = g(x) \cdot f(x)$  can generate  $2^{n'} - 1$  distinct patterns, provided that  $p(x)$  is a primitive polynomial of degree  $n'$ ,  $g(x)$  is any monic polynomial of degree  $n - n'$ , and that the initial state the LFSR is divisible by  $g(x)$ . For example, in a circuit with  $n = 4$ , if the maximum number of inputs upon which an output depends is  $n' = 3$ , then the characteristic polynomial of the condensed LFSR will consist of a primitive polynomial of degree 3, like  $f(x) = x^3 + x + 1$ , and the monic polynomial of degree 1,  $g(x) = x + 1$ . Hence, the characteristic polynomial in this case is,  $p(x) = f(x) \cdot g(x) = (x^3 + x + 1)(x + 1) = x^4 + x^3 + x^2 + 1$ . The corresponding IPG and its set of  $2^3 - 1 = 7$  distinct patterns are appear in Figure(2.4). From the figure, the pseudo-exhaustive property of having  $2^3 - 1$  non-zero patterns on any combination of  $n' = 3$  output lines is apparent. This technique is most efficient in regard to the test length when the maximum number of inputs driving an output is close to  $n$ . However, in general, it results in larger test sets compared to the other techniques.

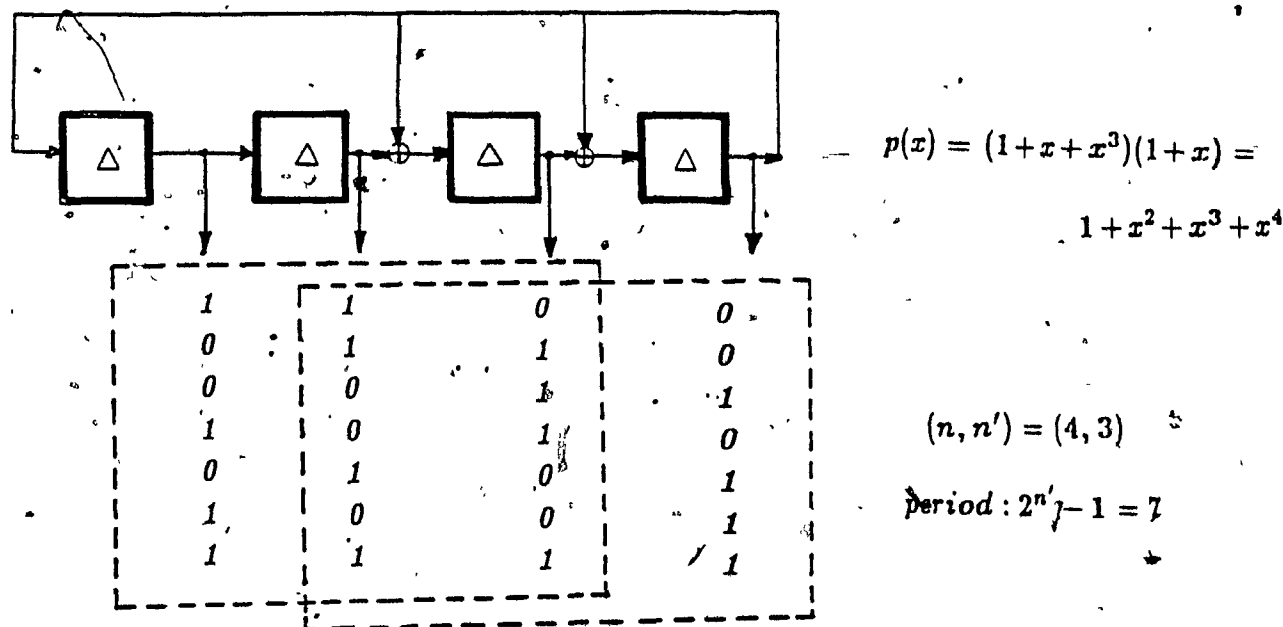


Figure (2.4) Pseudo-Exhaustive Input Pattern Generation

The use of linear sums [Akers 85] or linear codes [Vasanthavada 85] have been proposed to solve the test length problem where a combination of LFSR's and Exclusive-OR gates are used for input pattern generation. Yet another technique aimed at solving the same problem suggests the design of an LFSR-based on cyclic codes [Wang 86b]. The procedure to determine the characteristic polynomial for the latter LFSR is to first find a generator polynomial with specific degree  $k$  and design distance  $d$  from any coding theory text such as [Peterson 75]. The following step is to generate a  $(2^{\lceil \log n \rceil} - 1, k)$  cyclic code. The thing is to finally shorten the code to a size  $(n, k)$  cyclic code. This technique yields test lengths very comparable to those obtained in [Akers 85] and [Vasanthavada 85], but has lower hardware overhead requirements. Moreover, although verification testing [McCluskey 84] uses fewer patterns, [Wang 86b] claims that his technique is more attractive because of its easier design.

From all the above pseudo-exhaustive testing techniques, it seems that there is no single solution to the problem of generating pseudo-exhaustive input patterns that can simultaneously keep the test length, the number of initial patterns, and the hardware overhead to a minimum. One advantage of pseudo-exhaustive testing is that, in general, it guarantees single stuck-at fault coverage without any detailed circuit analysis [Archambeau 84]. Unfortunately, there are many circuits for which the pseudo-exhaustive testing techniques does not result in a satisfactory test procedure, because the resulting test set still remains too long to be feasible. Another alternative to exhaustive testing for BIST applications, that uses smaller number of test patterns, is the pseudo-random pattern generation.

### 2.2.2 Pseudo-Random Test-Based Techniques:

Random testing has been a common practice in industry for a long time. It has been observed that, for most combinational circuits, an effective fault detection is obtained if a sufficient number of randomly generated patterns are applied [Agrawal 75] [Schnurmann 75] [Malaiya 84] [McCluskey 85a].

Since there exists no ideal random number generator, it is not possible to obtain a truly random test set. However, as indicated earlier, it is possible to generate pseudo-random test sets that are considered good approximations to truly random ones. A major difference between random and pseudo-random test sets is that the latter is repeatable while the former is not. In reality, since testing requires test sets to be repeated, pseudo-random test sets are the suitable ones.

For a given circuit under BIST, the set of pseudo-random patterns which provides a desired fault coverage is usually larger than the set of deterministic patterns which

result in a comparable coverage, but is usually much smaller than the set of exhaustive patterns. One reason why pseudo-random pattern generation remain attractive in spite of the potentially larger test sets is that their *BIST* implementation can be provided by a non-expensive hardware device: the autonomous maximum cycle length *LFSR*.

Most of the *BIST* schemes with pseudo-random testing suggest using output data compression to reduce the large volumes of output data. The performances of different compression techniques are investigated in [Parker 76] and [Losq 78]. The form of *BIST* that combines pseudo-random pattern generation with polynomial division-based compression, being the most commonly used [Konemann 79] [Fasang 80] [Mucha 81] [Bardell 82], is considered to be the standard form of *BIST*. Few *BIST* schemes consider using pseudo-random testing without output data compression. For instance, [David 76] and [Shedletsky 75] suggest applying pseudo-random patterns to both a reference unit and to the *CUT*, considering the reference unit as a "golden" unit which provides the correct responses to the input patterns.

The major current issues in pseudo-random testing are establishing the test length, determining the fault coverage, and finding methods to detect the random pattern resistant faults. The solutions to all these problems could, in principle, be provided by performing fault simulation of the *CUT* [Waicukauski 85]. However, different less expensive methods have been developed to deal with these issues. Some of these alternatives are addressed in the two following sections.

#### **2.2.2.1 Establishing Pseudo-Random Test Length:**

The establishment of the pseudo-random test length for a given *CUT* is based on the estimated effectiveness of the pseudo-random patterns to detect the faults of the *CUT*,



as well as the prespecified fault coverage. One measure of detecting the effectiveness of pseudo-random patterns is the detection probability profile of a circuits' faults [Malaiya 84]. The detection probability of a fault is defined as the probability that a random test pattern detects the fault.

Different analytical methods that avoid the need for fault simulation have been set to estimate the test length. These are mainly based on algorithms that compute fault detection probabilities. Examples of such algorithms are: the Cutting algorithm, which estimates detection probability bounds [Savir 83a]; Predict, which uses a graph approach to compute exact probabilities instead of estimating bounds [Seth 85]; Stafan, which uses a statistical approach for the same purpose [Jain 84]; and finally, COP, which calculates the signal probabilities by assuming signal independence and thus ignoring reconvergent signals [Brglez 84].

Most of the methods for the test length establishment first find out the detection probability of the least detectable fault(s) of a CUT using one of the above algorithms. Given this probability the number of patterns necessary to achieve the desired test quality is predicted [Savir 83b], [Shedletsky 75], [Shedletsky 77], [David 76] and [Losq 78]. Other methods establish their predictions on the exponential relation between test lengths and expected fault coverages [Williams 85] [McCluskey 87].

The model of test pattern generation considered in most of the above mentioned methods the random process and not the pseudo-random process. The random model corresponds to sampling with replacement, thus each input pattern always has the same probability of occurring. It neglects the fact that patterns generated by an LFSR do not repeat until all of the non-zero patterns have occurred (i.e. sampling without replacement). [Chin 87] and [Wagner 87] show that it is inaccurate to establish the length of a test set by modeling the LFSR as a truly random source, because for a given

coverage it results in a larger number of patterns than a pseudo-random one. Methods to calculate test lengths by using the pseudo-random model are presented in [Malaiya 84] [Chin 87] [Wagner 87] [McCluskey 87].

Another shortcoming of the above mentioned methods is that the test length calculations consider only one factor, viz. the random pattern testability of a CUT. However, a second factor, viz. the effect of error information loss due to output data compression, must also be taken into account in establishing the test length. [Ivanov 87] analyzes this factor and suggests a method which adjusts the test lengths accordingly.

#### 2.2.2.2 Random Pattern Resistant Faults:

One of the major problems with BIST schemes based on pseudo-random testing is that certain faults resist detection by random patterns [Eichelberger 83]. These faults are known as random pattern resistant faults or hard faults [Savir 83a]. The difficulty in detecting such faults may result, either from the low probability of the faulty nodes to be randomly set to a logical 0 or 1, from the low probability of observing the logical values on these nodes at the circuit outputs. Hence, very few random patterns can both provoke such faults and sensitize them to an output. In other words, there is a very low probability of applying a random pattern capable of their detection [Eichelberger 83].

Several methods have been proposed to deal with random pattern resistant faults. One of these methods suggests finding specific test patterns to detect specific faults using a deterministic testing algorithm, and then add these new patterns to the test set [Savir 83a].

Another method suggests modifying the CUT by adding internal circuit nodes that are controllable and/or observable, to increase the probability of provoking and sensi-

tizing hard faults [Eichelberger 83]. This method adopts analytical testability measures [Savir 83a] [Agrawal 82] [Goldstein 79] to determine when and where logic modifications are necessary.

A more recent method to detect random pattern resistant faults proposes techniques to generate circuit-specific sets of pseudo-random patterns. Such circuit-specific sets are composed of unequiprobable (biased) patterns instead of the usual uniformly distributed patterns (equal probabilities of 1's and 0's). The change in the probability distribution is intended to bias the new set of patterns towards the few that detect the random pattern resistant faults. Such techniques are described in [Schnurmann 75] [Chin 84] [Wundelich 85] and [Wundelich 87]. They generally require the regular LFSR (IPG) to have additional circuitry, or additional circuitry to be inserted between the LFSR outputs and the CUT inputs. This circuitry serves to increase the frequency of occurrence of one logic value (0 or 1), while decreasing that of the other. Along the same lines is the adaptive input pattern generation technique proposed in [Timoc 83]. This technique assigns weights to the outputs of the LFSR such that a maximum fault coverage can be attained with a minimum test set.

It should be noticed that all the above mentioned methods adopt a circuit-specific strategy to improve the test quality of pseudo-random BIST schemes.

### 2.3 Output Data Compression Process:

Data compression is an operation used to reduce an amount of data to render its storage and analysis more economical. It has been established, that this operation is essential in a BIST design to economically handle the large volume of output data [Williams 84] [McCluskey 85a] [David 86]. Under most BIST schemes, output data

compression is used. In these cases, an ODC uses a function that maps the large amount of output data into a relatively short word. The compression process enables the reference data to be reduced to a word of identical size. In the following, the mapping function is referred to as a compression function, whereas the resulting short word as a signature.

In general, a signature is an attribute of the output data. Therefore, a compression function is usually chosen such that the signature depends on the entire output data. A compression function  $f$ , in general, can be thought of as many-to-one function when it compresses an  $l$ -bit long output data stream  $a_0$ , in the case of a single-output CUT, to a signature  $f(a_0)$ , or when it compresses an  $l \times m$  bit output data matrix  $A_0$  to a short signature  $f(A_0)$ , in the case of a multi-output CUT.

Under BIST, an ideal compression function is one through which all erroneous output responses always yield a faulty signature. However, no such function exists because any reduction in the amount of output data by compression causes some error information loss. This loss manifests itself in a number of erroneous output responses to map into signatures identical to the signature of the fault-free case. The mapping of erroneous output responses to signatures identical to the fault-free case prevents such faulty responses to be detected. The best result one can hope to attain from a compression function is to produce the least number of such mappings.

In a BIST scheme, the choice of a compression function is not only influenced by the error information loss, but also by another important constraint: the amount of additional hardware needed to implement this function. Usually, a compression function is not specifically designed for a given CUT, but is chosen from a set of easily realizable devices, without taking into account any information about the CUT. However, as it

will be shown in section 3.4, the new ODM scheme proposes an improved compression function which considers the CUT's function under a given set of input patterns. This consideration of functionality and input patterns is in fact equal to considering the resulting output response. In essence, a compression function that considers the output response is a circuit-specific compression. In this ODM scheme a circuit-specific compression is used to optimize the error information loss as well as the hardware overhead.

For compression functions to be easily implemented, they must be of low complexity. [Carter 79] first described a large set of functions from which to select compression functions. This is the universal set of hash functions which consists of the functions whose signatures all have the same probability of appearance. Then, [Wegman 81] suggested a comparatively smaller set for the same purpose. However, selecting a compression function from such large sets can often be inadequate, since more bits are required for selecting the actual function than to store the entire set of output data. Hence, most of the existing BIST schemes do not select their compression functions from such sets. Instead, they select their functions from easily realizable devices that perform compression with some sort of functional dependence on the entire output data. Several data compression devices, originally developed in connection with portable testers, serve as useful tools to realize this purpose and thereby reduce the large volume of output data [Hayes 76a] [Losq 78]. Examples of the most popular ones in the BIST area are LFSRs, parity trees and counters, whose compression techniques are respectively based on polynomial division, parity checking and counting. The remainder of this chapter addresses these techniques; whereas the considerations about their error information loss will be discussed in the next chapter.

Most of the BIST schemes use a compression technique to obtain a short (k-bit

long) signature from an (1-bit long) output data string of a single output CUT, or from an ( $l \times m$  bit) output data matrix of an  $m$ -output CUT. However, there are also a few BIST schemes which (i) either do not adopt data compression, or (ii) simply adopt it to map an  $l \times m$  output data matrix into an 1-bit serial stream with the same length. The no-compression case (i) happens when a BIST scheme needs to examine each output pattern concurrently. Such schemes are known as concurrent or on-line BIST schemes. Examples of such schemes are described in [Sedmak 79] and [Sedmak 80], where the functional circuitry is duplicated redundantly for concurrent checking. In regard to case (ii), termed as parallel to serial compression (P/S compression), such a case is required if a serial stream either is examined by a source outside the chip [Saluja 83] [Reddy 85], or compressed further during the self-test mode by another compression function. Examples appeared in [Zorian 84] [Zorian 86b] and more recently in [Robinson 87] and [Li 87].

After the compression step, the final operation of a self-test mode is to verify the correctness of the obtained signature. To perform this operation, it is necessary to determine the fault-free value of this signature in advance, based on the functionality of the CUT and the specifications of its BIST structure. This value is obtained either from a simulation of the fault-free circuit, or from an actual circuit verified to be fault-free by some other means.

### 2.3.1 Polynomial Division-Based Techniques:

It has been shown at the beginning of this chapter that an LFSR is the ideal device to perform polynomial division for output data compression [Benowitz 75] [McCluskey 86]. This polynomial division-based compression technique is also called signature analysis, a term coined by Hewlett-Packard [Chan 77] [Fröhwerk 77], to describe its use

in their products. The usefulness of this technique is due to the fact that the final value of a signature is an attribute of the entire output data stream applied to the LFSR's input. Figure(2.5) illustrates this fact with a simple example. Suppose a number  $l = 16$  of input patterns is applied to a single output CUT, and its resulting output stream  $a_0 = 1000110110100110$  is mapped by an LFSR of size  $k = 4$  to a signature  $f_{\text{lfsr}}(a_0) = 0110$ . Actually, this signature is a representation of the entire  $a_0$ , due to the continuous accumulation of data in the LFSR caused by the feedback connections. However, like any other compression device, an LFSR loses some error information. Hence, it is possible that a certain fault causes the output data to change to a different stream  $a_f \neq a_0$  which produces the same signature as the fault-free one, i.e.  $f_{\text{lfsr}}(a_f) = f_{\text{lfsr}}(a_0)$ .

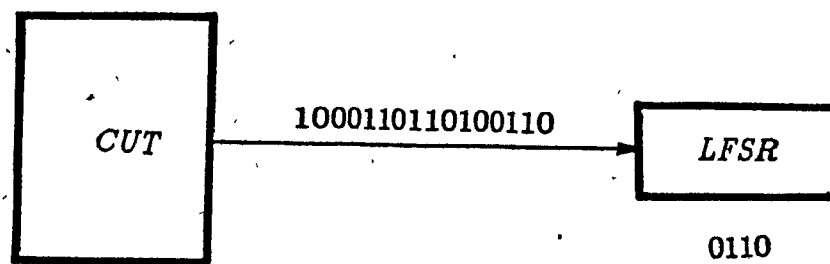


Figure (2.5) An LFSR as an ODC

The major deficiency of an LFSR adopted for polynomial division is that it has a single input. Therefore, it can only compress an output data stream but not a matrix. In general, since CUTs have multiple outputs, using the LFSR, for compressing the streams of every output would require a repetition of the test set as many times as the output multiplicity. Thus, if the test set is quite long, this deficiency results in an undesirable time overhead. To alleviate this problem, a multi-input linear feedback shift register, denoted as MISR [Benowitz 75] [Konemann 79], has been suggested. An

MISR is basically a polynomial division-based ODC where the serial data from the outputs of a CUT is fed through the Exclusive-OR gates between any two consecutive memory elements of the MISR, as shown in Figure(2.6). Thus, a new state can be processed within a single clock cycle. If the MISR is sufficiently long (number of stages) to accommodate all the output lines of the CUT, then the test set can be applied only once to determine whether or not the circuit is fault-free. However, even if the length of the MISR is less than the number of output lines, it is still possible to apply the test patterns only once by adopting a special configuration such as the one described in [David 86].

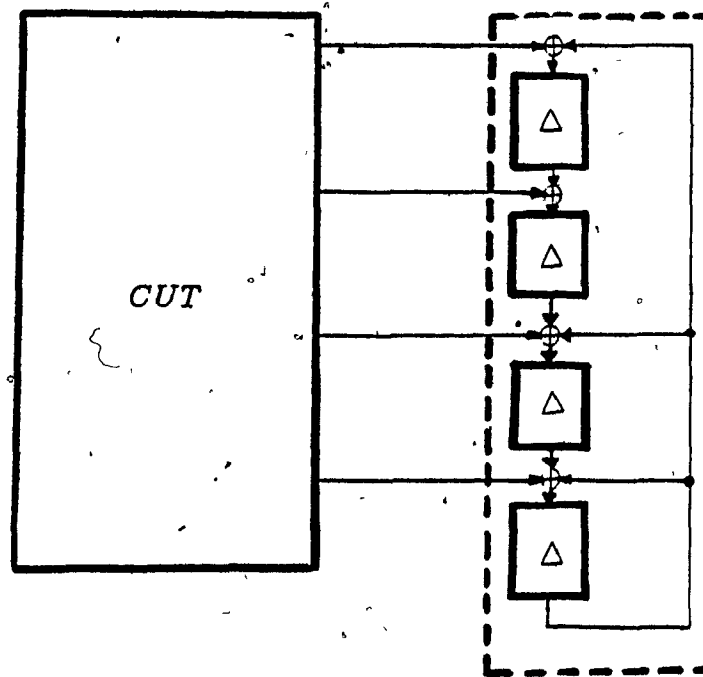


Figure (2.6) Multi-Input Shift Register MISR as ODC

From Figure(2.6) it is apparent, that an MISR requires more hardware than an LFSR of the same size. This is due to the additional inputs and larger Exclusive-OR gates. Furthermore, besides requiring extra circuitry, the MISR has an additional source of information loss known as error cancellation [Sridhar 82] [Hassan 83]. Such loss occurs



when an erroneous bit, from an output pattern, at a certain output line is followed by another erroneous bit, on the next pattern (following clock cycle) at the next output line. This sequence of events results in the latter erroneous bit to cancel the one that formerly entered the MISR. Both errors thus canceling each other, they have no effect on the signature.

Polynomial division (implemented by MISRs) used for output data compression are mostly found in the standard BIST schemes which also utilize LFSRs to generate their pseudo-random input patterns. The first such scheme appeared in [Benowitz 75], and then different variants followed. One of the most popular ones is described in [Konemann 79]. The latter scheme is most suitable for CUTs that can be partitioned into modules because the input and output registers of the resulting modules can be utilized for self-test purposes. To be more specific, this scheme suggests that an existing input register in a module first be reconfigured to an IPG which applies its pseudo-random test set to the module. Then, the same input register is reconfigured into an MISR which compresses the output data coming from its previous module. Such reconfigurable registers are called BILBO, i.e. built-in logic block observer. This scheme requires that consecutive modules be tested alternately [Konemann 79] [Konemann 80]. However, such an alternation lengthens the total test time. However, if the test time is a critical parameter for certain circuits, an improved BILBO scheme can be adopted [Wang 85]. This improved scheme reduces the previous test time by one-half by adopting a combined register called concurrent BILBO (CBILBO). CBILBO generates input patterns and compresses output responses simultaneously during the self-test mode.

A different scheme, shown in [Heckleman 81], suggests adding the standard BIST facilities to the CUT, instead of reconfiguring the existing registers as in the previous case. The self-test facilities in [Heckleman 81] consist of regular cells designed to form

IPGs or ODCs. Hence, such a scheme can be implemented with little effort due to its regular internal structure. More schemes, that use the standard form of BIST built inside the chip are described in [Eiki 80] [Fasang 80] [Mucha 81] [Komonytsky 82] [Resnick 83] [El-Ziq 83] and [Butt 84].

Other than the standard forms of BIST that are built inside the chip, there are the standard forms which maintain a chip design as is, and place the self-test facilities externally. Examples of such schemes are found in [Perkins 80] [Bardell 82] and [Bhavsar 85]. The last two references use a special test chip which contains both IPG and ODC.

In all the above mentioned BIST schemes, the signature obtained as a result of polynomial division-based compression is the ( $k$ -bit long) remainder of the polynomial division operation. However, in a few other proposed schemes, another polynomial division product is used, viz. the ( $l$ -bit long) quotient string. As its name implies, this string is the quotient of the polynomial division that appears at the output of the ODC. In some of these schemes [Sridhar 82] [Hassan 83], the entire quotient string of an MISR is verified for correctness. Such schemes therefore need to store an  $l$ -bit long reference value. This, in general, is a very expensive requirement for a BIST scheme. In another scheme [Zorian 84], which is our ODM scheme, the  $l$ -bit long quotient string is not verified until it is compressed into a signature of length  $k = \log(l)$  bits.

### 2.3.2 Parity Check-Based Techniques:

A simple parity checking operation on an output data stream detects a fault if the number of erroneous bits on the output stream is odd. This operation can be realized, as shown in Figure(2.7), by a parity checker composed of an Exclusive-Or gate and a memory element. In [Benowitz 75], the parity checking technique for output data

compression is discussed and [Benowitz 75], and compared with the polynomial division technique for pseudo-random input patterns. [Benowitz 75] claims that if a CUT can be partitioned such that its outputs or test points are grouped by their association with common logic, and each output of a group connected to a different parity checking ODC, then this technique detects 95% of the single faults propagated to the ODC.

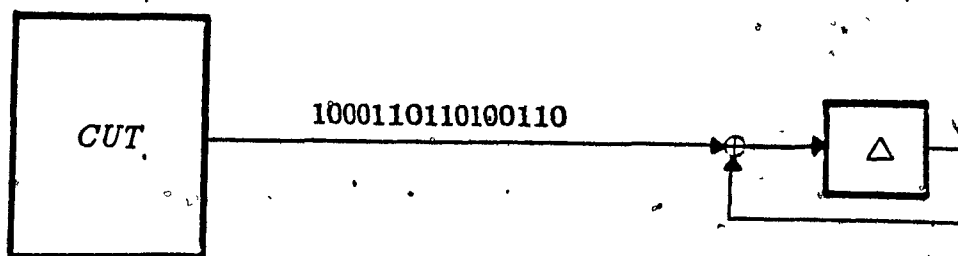


Figure (2.7) Parity Checker as ODC for an Output Data Stream

If an  $m \times l$  bits output data matrix is compressed by parity checking, the ODC may be composed of  $m$  parity checkers, one on every output line. This provides an  $m$ -bit long signature. Another ODC which performs a parity check-based compression is a parity tree. Such tree is equivalent to an Exclusive-OR gate with  $m$ -inputs which performs parallel to serial compression (P/S compression) by mapping the output data matrix into a single stream of length  $l$ . Both parity check-based ODCs are used by existing BIST schemes [Benowitz 75] [Carter 82b] [Robinson 87].

Useful analyses of the utilization of parity trees in BIST are provided in the literature [Saluja 83] [Reddy 85], and a complete BIST scheme that uses parity tree as ODC is found in [Carter 82b] [Carter 82c]. This scheme utilizes exhaustive test sets as input patterns, and a number of parity trees with an accumulator for output data compression. More specifically, it divides the CUT outputs into subsets, and combines the lines of each subset by means of a parity tree. The parity tree outputs are then connected to an accumulator to calculate the sum of the parities for each input pattern. Although

this scheme yields good stuck-at fault detection and is fast; its implementation is more costly than other schemes, and more importantly, it causes the worst error information loss [Wang 85].

Several other BIST schemes also suggest using parity check-based compression. One example is found in [Davidson 81] where the Bellmac-32 microprocessor chip uses parity checking combined with polynomial division. Another example is demonstrated in [Akers 86] where exhaustive testing with a special parity bit signature are adopted. In [Tzidon 78], parity checking is used to accomplish the detection of all multiple stuck-at faults in tree networks. Finally, [Bhattacharya 87] developed an existing testability analysis method to analyze the parity testability of a CUT, and also suggests modifications to make it testable. In section 3.4, the effectiveness of compression by parity checking will be discussed, along with the effectiveness of other types of ODCs.

### 2.3.3 Count-Based Techniques:

A count-based compression technique consists of counting a certain attribute of the output response so that the signature obtained at the end of the self-test procedure represents the final count of that attribute. Various such techniques are used in the existing BIST schemes. In fact, the common characteristic among these techniques is that their ODCs are merely composed of a counter in some cases preceded by a digital function [Parker 76].

The simplest and better known examples of count-based techniques for single output circuits are: One's Counting which counts the number of logic ones in an output data stream being compressed [Hayes 76a] [Savir 80] [Agarwal 83]; Transition Counting which sums the number of times the output data stream changes values (from 1 to 0

and from 0 to 1) [Hayes 76b] [Reddy 77] [Hayes 78]; and *Edge Counting* which counts either the changes from 1 to 0 or the changes from 0 to 1 [Fujiwara 78] [Parker 76]. The realization of the three previous count-based techniques are illustrated in Figure(2.8).

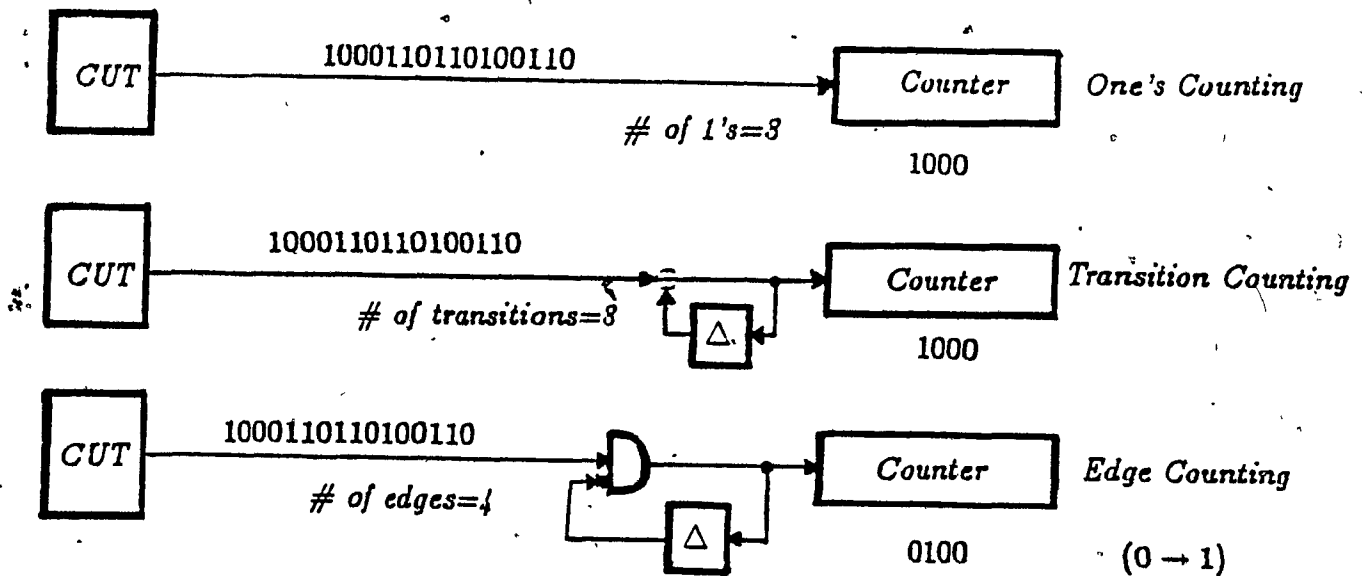


Figure (2.8) Count-Based Compression Techniques for Single Output CUTs

All of the three count-based techniques provide a logarithmic compression of data. In other words, with those techniques, the length of a signature is usually proportional to  $\log(l)$ , where  $l$  is the test length. However, the length of a signature can be reduced further if one utilizes the dependence of the signature upon the order of input patterns. Such a dependence does not exist in the case of One's Counting, while in the Transition and Edge Counting, it can be seen that a signature is dependent on the order of input pattern application. Thus, in such cases, it is possible to minimize the number of transitions or edges in the fault-free output data by ordering the patterns in advance. Such ordering, in the extreme, results into a single 0 to 1 (or 1 to 0) transition [Fujiwara 78].

One of the well known BIST schemes which utilizes a count-based compression technique is called syndrome testing [Savir 80]. This scheme uses the One's Counting technique coupled with exhaustive testing. Hence, the signature that it provides at the end of the counting procedure is the function's weight (i.e. the number of its minterms). It has been shown that it is possible to detect any single stuck-at fault in a CUT using this scheme [Savir 81]. However, since certain faulty circuits produce signatures identical to the fault-free one, specific CUT modifications are generally required to ensure the detection of the faulty circuits, i.e. to produce syndrome testable designs [Savir 81] [Markowsky 81]. Producing a syndrome testable design, in general, implies adding extra I/Os and gates. Another drawback of syndrome testing is that since exhaustive test sets are adopted to reduce the test duration to an acceptable level, the difficult problem of CUT partitioning is raised. Furthermore, if syndrome testing is applied to multi-output CUTs, it has been suggested to replace the single input counter by a very expensive multi-input ODC known as weighted syndrome counter [Barzilai 81]. One implementation of a weighted syndrome counter consists of an adder and a register (Figure(2.9)), both of length  $m + n$ , where  $m$  is the number of CUT outputs and  $n$  is the number of inputs. In addition to being expensive to realize, the One's Counting technique causes, on the average, substantial losses in error information [Hurst 87] similarly to the other count-based techniques. The error information loss is discussed further in section 3.4.

Another BIST scheme which uses count-based technique is presented in [Susskind 81], where the counting of Walsh coefficients for output data compression is introduced. Like syndrome testing, this scheme uses exhaustive testing and counts two out of  $2^n$  Walsh coefficients, one of them being the number of ones in a stream (i.e. same as One's

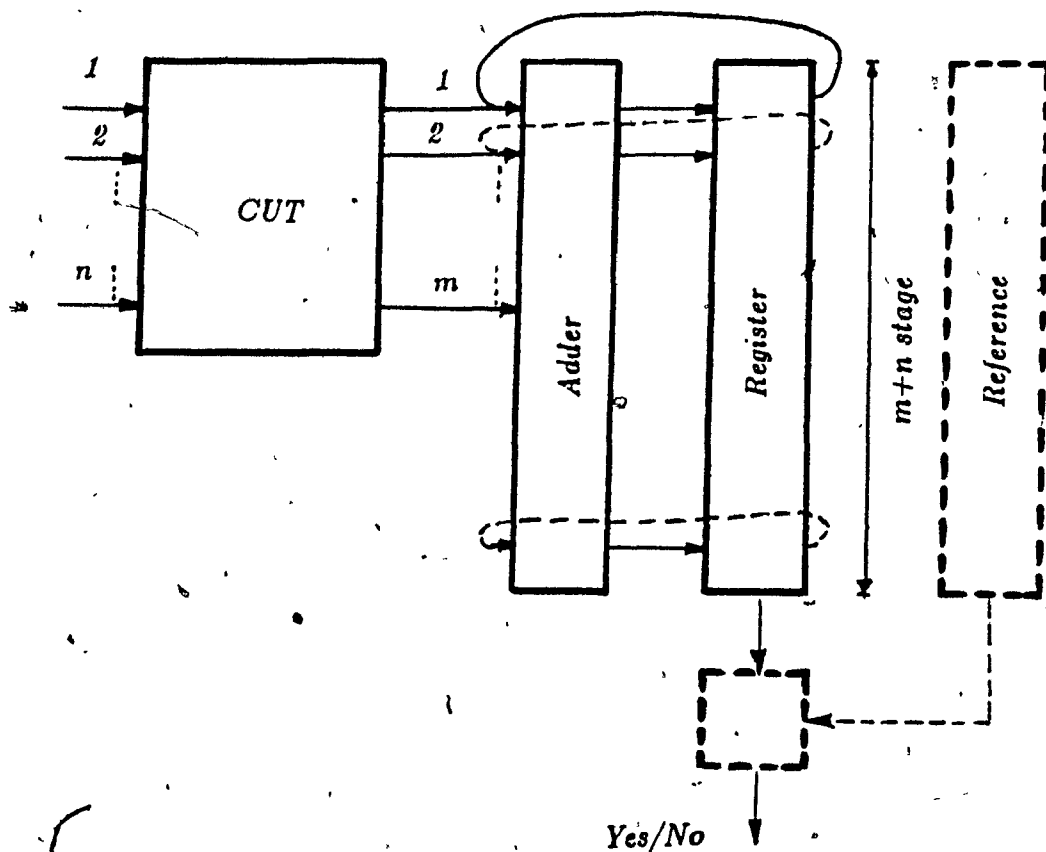



Figure (2.9) One's Counting for Multi-Output CUTs

Counting). If the implementation of the scheme is such that these two coefficients are counted serially, the test set has to be applied twice. In that case, the test time is doubled, and if they are done in parallel, two ODC counters are needed, one for each coefficient. Nevertheless, in either case, two reference values are required. Moreover, alike the syndrome testing, this scheme requires circuit modifications to cover all stuck-at faults.

An extension to the syndrome testing and walsh coefficient testing schemes is provided in [Muzio 83] and [Miller 84], where the count-based techniques are given a more general treatment by considering all possible spectral coefficients [Hurst 85]. A test in the latter scheme consists of verifying one or more spectral coefficients of the output data stream. The distinction with the other schemes is that instead of modifying the circuit to eliminate the syndrome untestable cases, the coefficients corresponding to



these cases are added to the set of spectral coefficients under verification. Although the difficult modification step of the previous schemes is avoided by selecting a circuit-specific set of coefficients, the resulting signature verification procedure, as well as the reference storage required, become more expensive in this case. Moreover, in the case of multi-output circuits the area-overhead expenses that walsh and spectral coefficients count techniques ensue are exacerbated.

Under all count-based techniques, the amount of loss in error information, for average cases, is considerably high compared with polynomial division-based techniques [Hurst 87] [Robinson 87]. In the next chapter, the calculation of such losses is illustrated, and a new concept which suggests utilizing a particular characteristic of the count-based techniques to reduce this loss is introduced.

Several new BIST attempts suggest the use of sophisticated compression techniques [Bhavsar 84] [Hassan 83] [Robinson 87] [Li 87]. These techniques, which are mainly intended to improve the test quality, are in fact extensions or combinations of the previously designed ones. The performance measurement of these various compression techniques, as well as the need for improving their test quality, is the main interest of the next chapter.



## Chapter 3

### Error Masking Problem and Output Data Modification

#### 3.1 Introduction:

The adoption of *BIST* as an alternative to the deterministic testing strategy of *VLSI* testing has been shown in chapter 1 to have certain crucial requirements. One of the most important requirements is to adopt an economical way to deal with the large amount of output data which results during the test procedure. In order to satisfy this requirement, various output data compression techniques have been developed, as seen in chapter 2. However, all of these techniques result in a loss of error information due to the reduction in the amount of output data. This loss of error information in output data compression is known to be a major cause of concern [Smith 80] [Sridhar 82] [Carter 82a] [Agarwal 83] [Bhavsar 84] since it causes a certain reduction in the number of detected faults, and hence, in the quality of test. An important aim is to

increase the quality of test in a BIST implementation without incurring a large overhead. Various attempts have been made to realize this aim and thus improve the test quality by reducing the amount of loss in error information [Hassan 84a] [Bhavsar 84] [Carter 82a] [Li 87] [Hurst 87]. Recently, the new output data modification (ODM) scheme [Agarwal 83], [Zorian 84] [Zorian 86a] and [Zorian 86c], which is the main subject of this dissertation, has been developed for the same purpose. This scheme provides the best improvement in the quality of BIST over all the other existing schemes by providing high fault coverage, with a small amount of increase in the area overhead. The improved test quality achieved with ODM is obtained by exploiting the functionality of the CUT. Consideration of the CUT's functionality classifies the ODM scheme under the circuit-specific BIST strategy.

This chapter introduces the output data modification concept and proposes a BIST model that uses ODM. Prior to these introductions, a proper description of the error masking problem and an illustration of the existing BIST schemes aimed at improving the error coverage are given.

To refresh the reader's memory on the notation previously introduced and that is still used in this chapter,  $l$  denotes to the number of input patterns applied to the CUT;  $a_0$  is the fault-free output stream of length  $l$  for the single output case;  $A_0$  is the  $l \times m$ -bit fault-free output matrix for the multi-output case, where  $m$  is the number of output lines. In general, the multi-output case is the one treated in the following.

### 3.2 The Error Masking Problem:

Since the reduction of output data by ODM, results in a loss of error information, certain errors are bound to go undetected. Such a phenomenon is called error masking

(or sometimes aliasing). More precisely, error masking occurs when a faulty and the fault-free CUT produce different output responses that have identical signatures. As a result, the faulty CUT is incorrectly concluded to be fault-free though it is defective. Numerous papers address the question of error masking under different output data compressors [David 76] [Segers 81] [Agarwal 83].

The detection of faults after compression is what is of primary concern. Unfortunately, one cannot directly observe faults after compression, nor can one generally afford to simulate the effect of every possible fault on CUT's output response and ODC. Nonetheless, one can more readily observe the effects of faults, i.e. the errors. Hence, a typical measure of masking is in terms of errors detected, i.e. error coverage (not fault coverage), usually determined by a probabilistic approach [Smith 80]. Having informally introduced the problem of error masking, we will now treat it in a more rigorous manner. We let  $T$  be the number of all possible  $m \times l$  matrices; and let  $A_0$  represent the output response of the particular fault-free CUT under consideration, and  $A$  be the set of all possible output responses (excluding  $A_0$ ) such that for each  $A_1 \in A$ , the signature  $f(A_1)$  is equal to the error-free signature  $f(A_0)$ , where  $f$  is the compression function. The number of elements in the set  $A$  is defined as the deception volume of  $A_0$  with respect to  $f$  and denoted by  $dv(A_0, f)$ . In other words, the deception volume is the number of output responses with the same signature as that of the fault-free output response, i.e. is the number of masked output responses.

If  $dv(A_0, f) = 0$ , then there is no error information loss, but if  $dv(A_0, f) > 0$ , then there are  $dv(A_0, f)$  possible erroneous responses which are undetectable by the testing scheme under consideration. Thus, the quality of an ODC, for a given CUT, may be measured by deception volume. Figure(3.1) illustrates a general mapping of the set of all possible output responses to the set of signatures accomplished by a compression function.

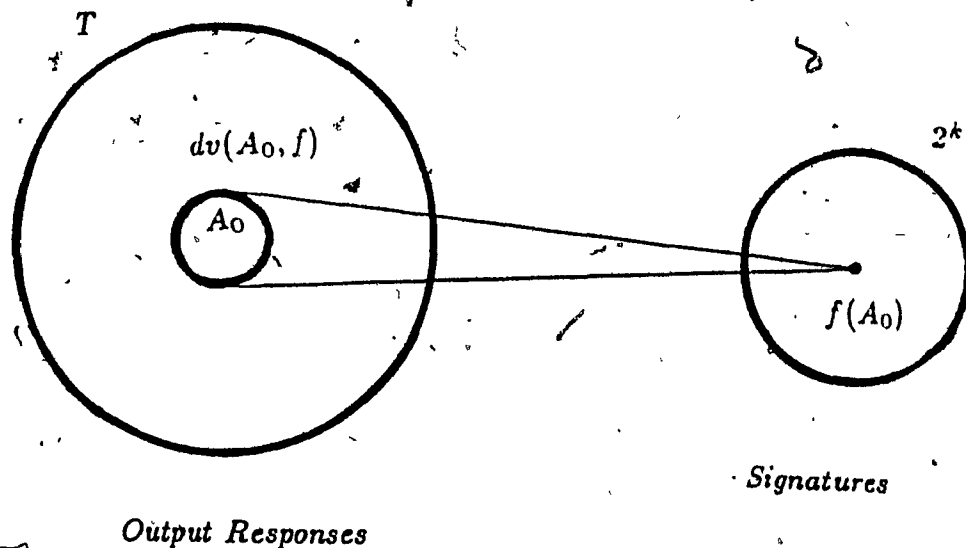


Figure (3.1) A Representation of an Output Data Compression

If each one of the all the possible output response matrices produced by faulty CUTs has the same probability of occurrence, then the ratio:

$$pr = \frac{dv(A_0, f)}{T}$$

is referred to as the error masking probability of  $f$ , given  $A_0$ . A small  $pr$  indicate an efficient  $f$  and/or  $A_0$ . Although it is known [Carter 82a] [Smith 80] that, in general, no CUT satisfies such an equi-likelihood assumption; however, since no other model that suitably explains the actual behaviors of faulty circuits have been developed to date, we also used the conventional assumption of equi-likelihood. The definition of  $pr$  permit values for it to be easily calculated. We will show examples of calculations for different ODCs.

In the case, of a  $k$ -bit long LFSR the compression involves only  $GF(2)$  additions which means that an equal number of output data streams will be mapped to the same  $k$ -bit signature [Smith 80]. Therefore, there are exactly  $\frac{2^l}{2^k}$  streams that map to the same signature. If there are  $T = 2^l - 1$  possible output data streams considered

erroneous, the deception volume of an LFSR is  $dv(a_0, LFSR) = 2^{l-k} - 1$ . Then the masking probability is  $pr = dv(a_0, LFSR)/(2^l - 1) \approx 2^{-k}$  [Frohwerk 77] [Smith 80]. It is shown in [Williams 86] [Williams 87] that it is possible to achieve a masking probability value of  $2^{-k}$  without making the equi-likelihood assumption. However, this  $2^{-k}$  value is only attained as the test length tends towards infinity.

In the case of an MISR, using the same arguments as for the LFSR, we find the deception volume to be  $dv(A_0, MISR) = 2^{l \times m - k} - 1$ , where  $k$  is the length of the MISR, and hence,  $pr$  is simply  $2^{-k}$  for large  $l$  and  $m$  [Bhavsar 81] [Sridhar 82].

As an example of the calculation of deception volume and masking probability for count-based techniques, we consider the specific case of One's Counting. To measure the deception volume in this case, consider the example shown in Figure(2.8). In this example, the output data string of length  $l = 16$  is compressed to the signature  $w = 8$ . But, there are other strings of the same length ( $l = 16$ ) that have the same number of ones; and thus result in the same signature  $w = 8$ , if compressed by the same technique. The number of such strings in this example is  $\binom{16}{8}$ . In general, the deception volume of One's Counting is  $dv(a, 1's) = \binom{l}{w} - 1$ , and its masking probability is  $pr = \frac{\binom{l}{w} - 1}{2^l - 1}$ .

In fact, the number of ones expected in almost all sequences of length  $l$  is close to  $l/2$ . Hence, the deception volume of such sequences is approximately  $\binom{l}{l/2}$ . It has been shown that for such typical sequences, more errors are missed by One's Counting than by a polynomial division-based ODC of the same length [Robinson 87]. It has also been shown that the error masking probability for One's and Transition Counting asymptotically approaches  $1/\sqrt{\pi l}$ , when averaged over all possible circuits and all possible errors [Savir 85]. Based on this probability, [Savir 85] also states that polynomial division-based compression reaches lower error masking probabilities than One's or Transition Count-based techniques for a given  $l$ . Furthermore, different experimental results also

demonstrates that One's Counting has a lower performance than polynomial division [Robinson 87]. Other count-based compression techniques, like Transition, Edge or Spectral techniques, are shown to be inadequate as well, if applied to average sequences [Hurst 87]. From the above findings concerning count-based techniques, the reason behind the popularity of polynomial division-based output data compression, and particularly the use of MISRs in the large number of schemes using the standard form of BIST is clear.

If  $k = 16$  in an MISR of length  $k$ , then generally  $pr = 2^{-k}$  is considered to be a reasonable masking probability. On the other hand, the realization that the denominator  $T$  used in calculating  $pr$  is an extremely large and unverified number, it appears reasonable to expect that for large and complex CUT's,  $T$  should be a much smaller number, i.e.  $pr$  be larger than  $2^{-k}$ . Several papers address the problem of reducing the error masking probability [Sridhar 82] [Agarwal 83] [Bhavsar 84]. Consequently, various BIST schemes to solve this problem have been reported in the literature [Hassan 83] [Carter 82a] [Hassan 84a] [Bhavsar 84] [Hlawiczka 81] and [Hlawiczka 86]. Most of them try to improve the error coverage of MISR-based compression schemes by suggesting more sophisticated compression techniques to reduce the masking probability in such schemes. Some of these attempts, along with the enhancements in error coverage they provide, will be considered in section 3.3.

In the following discussions, the two expressions decrease error masking and increase error coverage will be used without any distinction. Moreover, wherever the compression function  $f$  is obvious, we will write  $dv(A_0)$  instead of  $dv(A_0, f)$ .

### 3.3 Improving Error Coverage:

The effectiveness of an ODC is conditional upon its associated error masking probability. Ideally, it is desirable that masking probability be reduced to zero, but, in fact, this can only be achieved by abandoning output data compression and using the entire output response as the signature. In effect, masking is discouraging because it implies the loss of "hard-won" information. Even if the input patterns are good enough to detect a particular fault, if the erroneous output response is compressed into the same signature as the correct one, then the detection of this fault vanishes. Some of the different approaches aimed at reducing the error masking problem that have investigated several new compression techniques are reviewed in the following.

#### 3.3.1 Extended Compressor Approach:

Since the error masking probability is related to the number of stages in the signature register, the simplest approach is to increase the number of stages from  $k$  to, say,  $2k$ , as shown in Figure(3.2). With this, the probability of error masking becomes  $pr = 2^{-2k}$ . This reduction, however, is not very significant in regard to the reduction in deception volume. For instance, when  $l = 1000$  and  $m = 16$ , the deception volume with  $k = 16$  is  $2^{16000-16}$ , and with  $k = 32$  is  $2^{16000-32}$ .

Moreover, in regard to extra hardware, this enhancement requires  $k$  additional compressor stages, along with the corresponding storage required for the extra reference values.

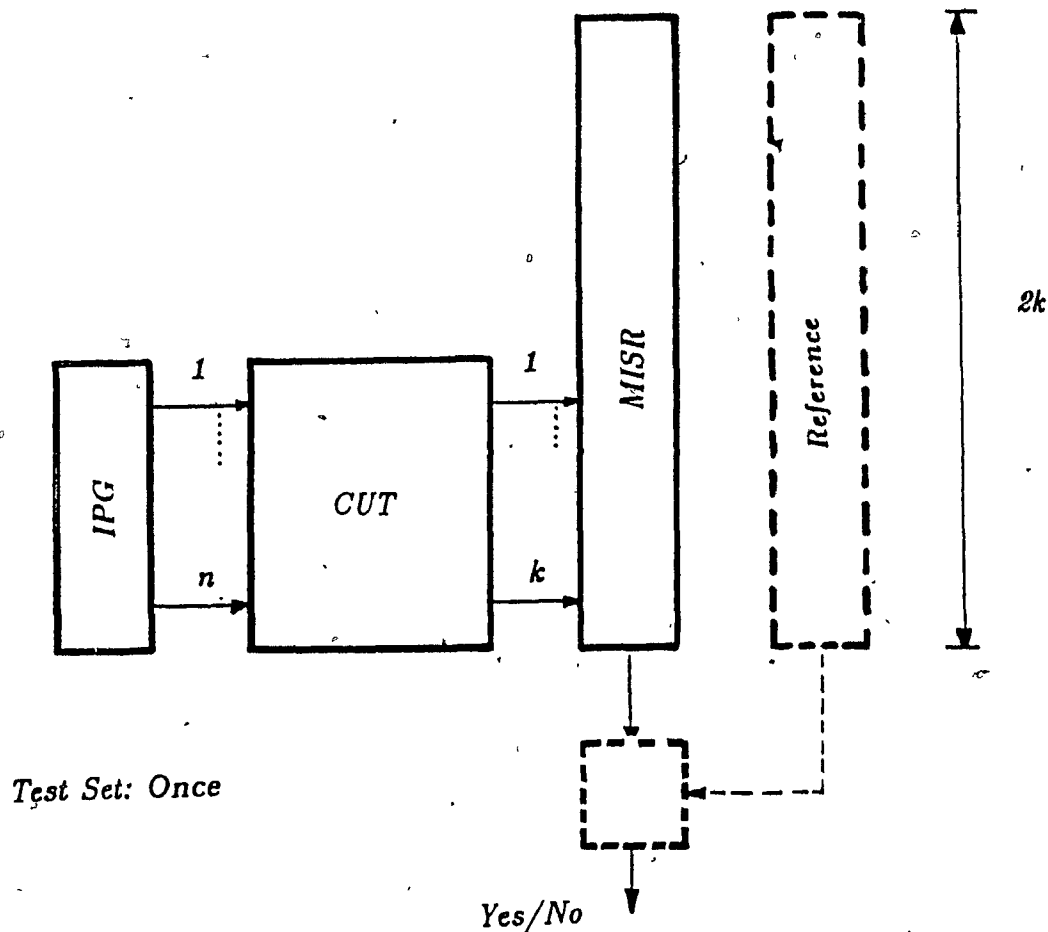


Figure (3.2) Extended Compressor Approach

### 3.3.2 Segmented Test Set Approach:

The Segmented Test Set Approach is one of the simplest enhancements of the standard MISR compression. In this approach, the original test set is segmented (subdivided) into two or more subtests, and a signature is collected for each subtest [Bhavsar 84] [Robinson 87]. The signature may or may not be collected independently of the others, that is, the MISR is either re-initialized only once, i.e. at the beginning of the test, or prior to each subtest.



Without loss of generality, let the original test set of length  $l$ , be segmented into two subsets, of length  $l_1$  and  $l_2$ , as shown in Figure(3.3). In this case, masking occurs when the signatures of both subsets are identical to the two corresponding fault-free signatures. Due to this segmentation, the deception volume is reduced to only those erroneous matrices which are not detected by neither the first nor the second parts of compression. The calculation of this reduced deception volume can be derived as follows. The number of matrices that produce the first signature is  $2^{l_1 \times m - k}$ , and the number producing the second is  $2^{l_2 \times m - k}$ . The total number of matrices, which produces both signatures is the product of the two previous numbers, i.e.  $2^{l_1 \times m - k} \times 2^{l_2 \times m - k} = 2^{l \times m - 2k}$ . Hence, the deception volume is  $2^{l \times m - 2k} - 1$ , and the masking probability in this case is reduced to  $2^{-2k}$ .

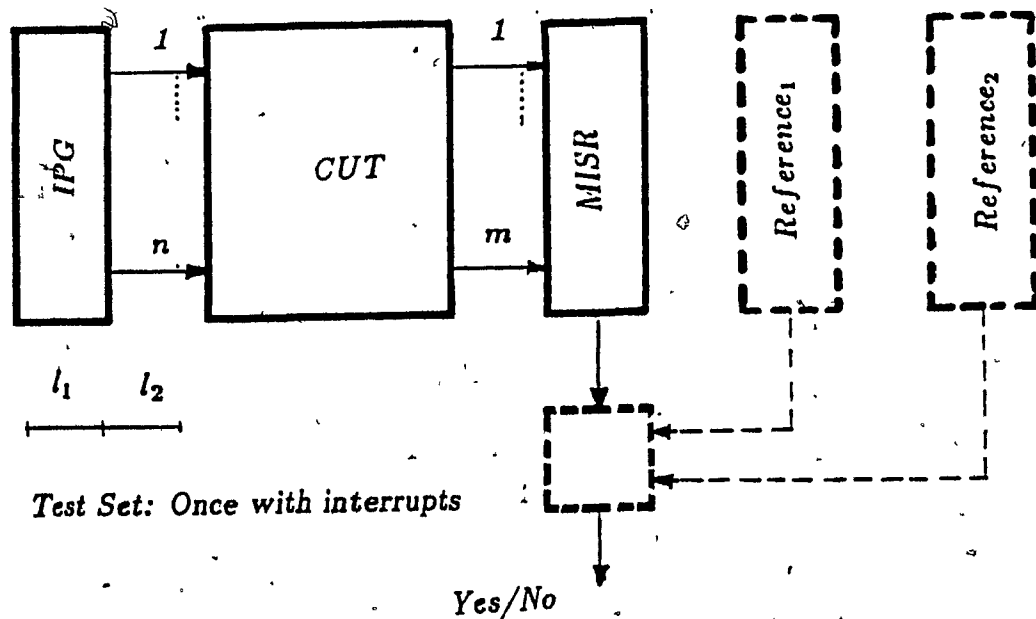


Figure (3.3) Segmented Test Set Approach

Note, that the above results are independent of where the test set is segmented. Furthermore, the hardware overhead in this approach is only in terms of storage needed for the reference values. However, the time to compare the signatures with the references

is longer in this approach compared with the previous approach.

### 3.3.3 Multiple Level Compression Approach:

Another approach [Hassan 83] suggests using multiple levels of compression to improve error coverage. This approach is illustrated in Figure(3.4), for the two level compression case. Assuming the CUT to have  $m$  output lines, the first level of compression consists of  $r$  MISRs each of  $m/r$ -bits long. The  $m$  output lines of the CUT are divided into subsets, and each subset is fed to one of the MISRs. Hence,  $r$  different signatures are obtained at this level. At the second level of compression, the  $r$  quotient-bit strings from the outputs of the first level MISRs are fed to a new  $r$ -bit long MISR to obtain an additional signature.

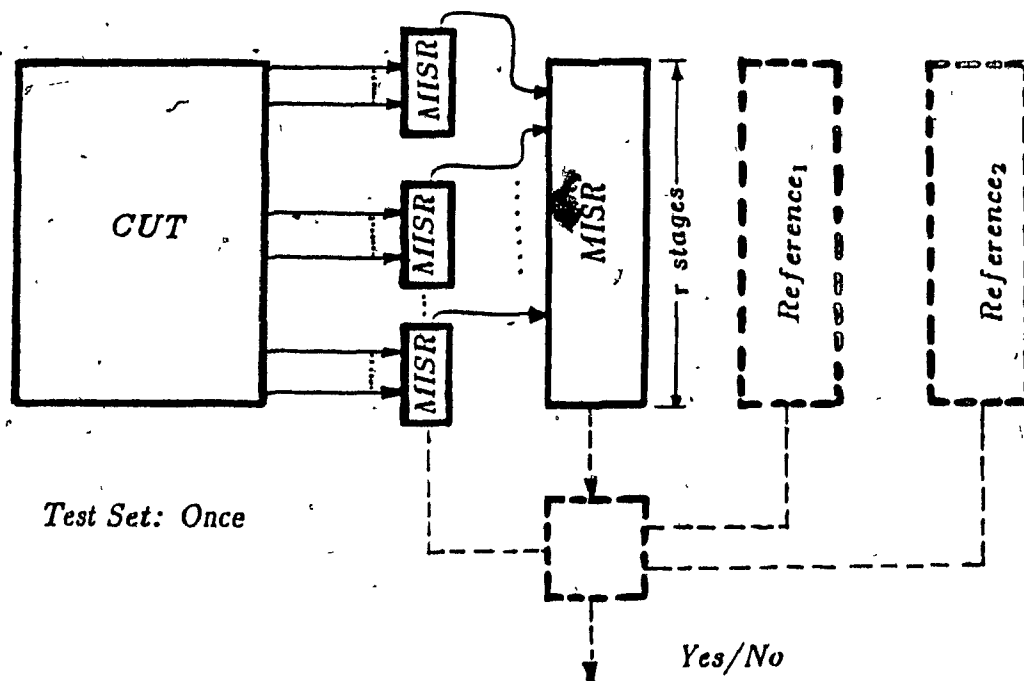


Figure (3.4) Multiple Level Compression Approach

In this type of output data compression, error masking may occur in the ODCs of both levels. The number of matrices which are mapped to the same signature in each MISR of the first level is  $2^{l \times \frac{m}{r} - \frac{m}{r}}$ . Hence, the total number of matrices which produce  $r$  fault-free signatures simultaneously is:

$$2^{l \times \frac{m}{r} - \frac{m}{r}} \times 2^{l \times \frac{m}{r} - \frac{m}{r}} \times \dots = 2^{l \times m - m}$$

Thus, the masking probability of the first level is  $pr_1 = 2^{-m}$ . In regard to the probability of error masking occurring at the second level MISR, it is simply  $pr_2 = 2^{-r}$ . Therefore, the probability of an erroneous matrix going undetected at both levels of MISRs is  $pr = pr_1 \cdot pr_2 = 2^{-m} \cdot 2^{-r} = 2^{-(m+r)}$ . In conclusion, for the same number of additional MISR stages, the reduction in error masking under the Multiple Level Compression Approach is comparable to that obtained for the Extended Length Compressor Approach.

### 3.3.4 Multiple Test Sets Approach:

Another approach for reducing error masking suggests applying multiple test sets [Hlawiczka 81] [Hassan 84a], each of which essentially detects the entire set of faults. Figure(3.5) illustrates this approach with two test sets,  $t_1$  and  $t_2$ . During the application of each test set, a signature is collected in the same  $m$ -bit long MISR and then compared with its corresponding reference value.

The implementation of such an approach requires no additional hardware on the ODC side. However, the duration of the entire test is multiplied. Moreover, it places the burden on finding non-expensive input pattern generation techniques to both generate and apply the test sets. One easy way of generating two distinct test sets for single

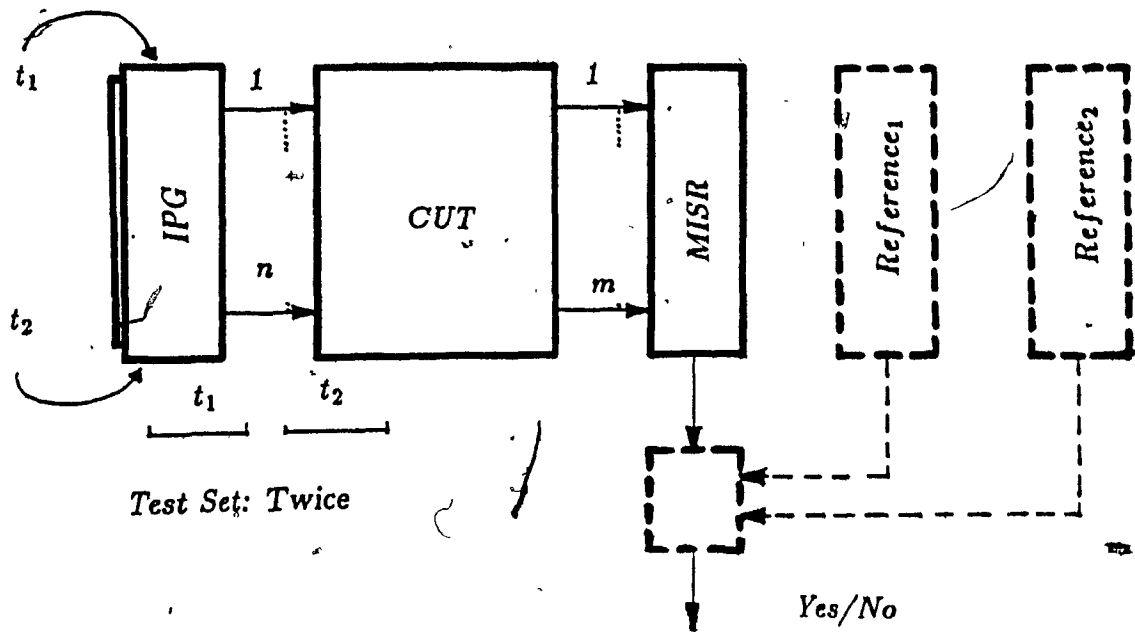


Figure (3.5) Multiple Test Sets Approach

output circuits is presented in [Hlawiczka 81] and [Hassan 84a], and its extension for multi-output circuits is described in [Hassan 84b] and [Hlawiczka 86]. The basic idea behind this particular technique is to use an original test set once, and then reverse its order of patterns to use it as a second set.

In the Multiple Test Set Approach, the deception volume consists of the erroneous output data matrices that are compressed into the fault-free signatures with each test set applied. It can be shown that by using two signatures, the error masking probability is  $pr = 2^{-2m}$ . Hence, the deception volume for this approach is reduced by similar factors as for the previous approaches.

### 3.3.5 Multiple Compressions Approach:

Another approach to reduce error masking suggests employing two or more MISRs, each having a different characteristic polynomial. Figure(3.6) shows an example where

two MISRs are employed. In a similar way, one MISR with two different characteristic polynomials can be used, repeating the same test set twice, trading off extended test time for reduced hardware. Another alternative is to provide an adjustable length MISR, e.g. of length  $k$  and  $k+1$ , with a facility for changing the length for the second application of the test.

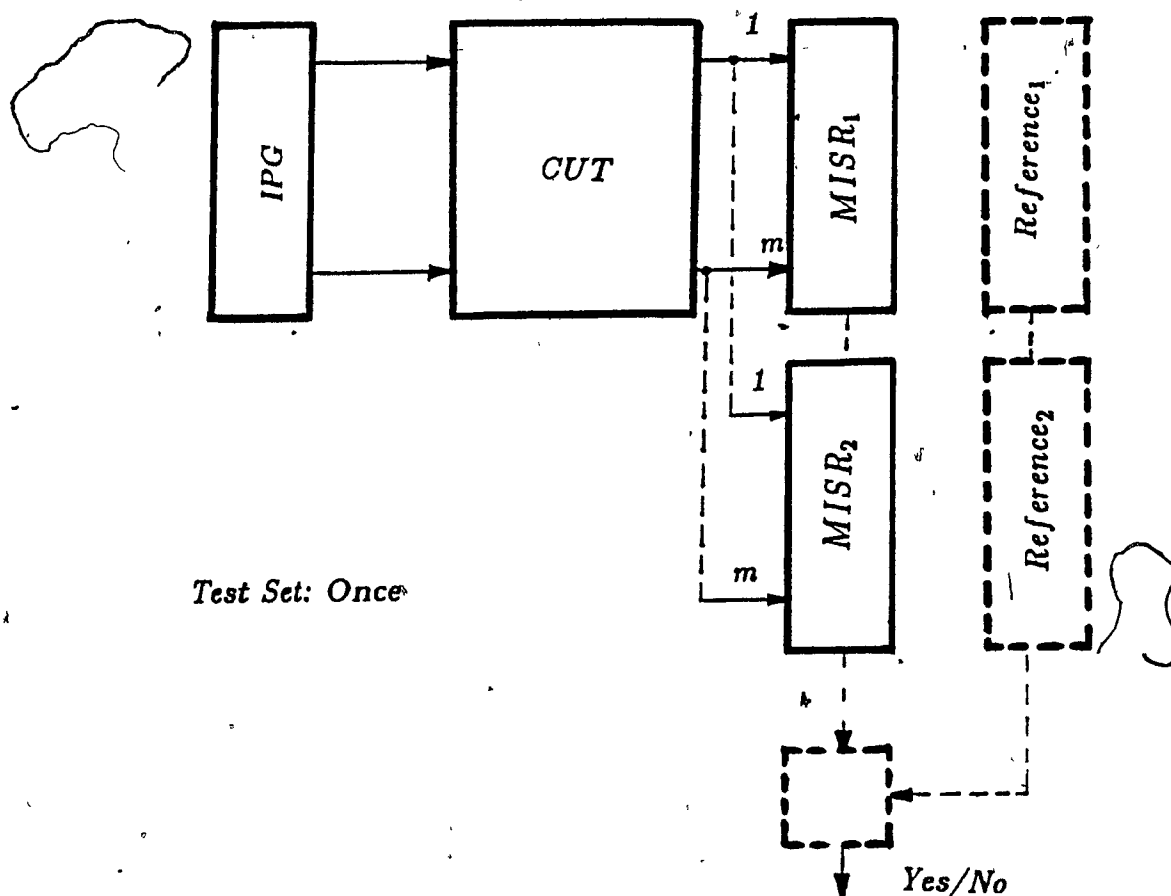


Figure (3.6) Multiple Compressions Approach

It has been shown [Bhavsar 84] that for any two characteristic polynomials of degree  $m$ , there exists  $2^{l \times m - 2m}$  matrices of size  $l \times m$ , which are divisible by both characteristic polynomials. Therefore, the reduced deception volume is the same as in the previous cases. Thus, the error masking probability is  $2^{-2m}$ . This approach has the same reduction in error masking for the same amount of extra hardware as the above

approaches.

### 3.3.6 Variable Shift Compression Approach:

Yet another set of suggestions to reduce error masking in polynomial division-based compression are made in [Carter 82a]. A result from the latter is that if the  $l$  input patterns in a test set are chosen by a truly random process, then the masking probability is bounded by  $4/l$ . However, it can be demonstrated that no pseudo-random number generator can be considered as truly random because of the correlations between the input patterns [Carter 82a]. The suggestion to reduce masking probability to  $4/l$  based on this fact is to randomly permute the output lines of the CUT before feeding them into an MISR. However, the required area to connect the outputs to the MISR in a random way is too great to be practical [Carter 82a].

A more practical approach called variable shift compression is suggested in [Carter 82a] as well. The idea in this approach is to reduce the risk of masking resulting from the fact that both input and output registers of the CUT are shifted simultaneously, one bit position per test cycle. The suggestion is to modify the MISR to permit either one or two shifting clock cycles between each output pattern from the CUT. In that case, the choice of a single or double shift can be made randomly by a signal on an extra control line. It is shown [Carter, 82a] that the masking probability in this case is no more than  $pr = 1/l$ . It is not possible to compare this masking probability with those of earlier approaches, because this approach does not make the equi-likelihood assumption as all the others do.

### 3.3.7 Combined Compression Approach:

The last approach for reducing masking considered in this section suggests us-

ing more than one type of ODC to reduce the deception volume. An example is described in [Robinson 87], where a count-based compression is added to the regular polynomial division-based one. As shown in Figure(3.7), the  $m$  output lines of the CUT are first compressed to an  $l$ -bit long string by an Exclusive-OR tree (parity tree). Then, this string is fed to both an LFSR as well as a counter to be compressed separately. This approach also yields two signatures, but utilizes three compression blocks.

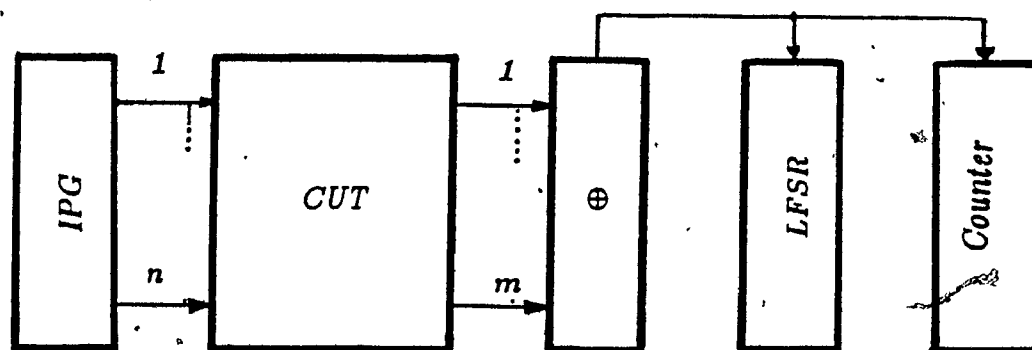


Figure (3.7) Combined Compression Approach

The additional hardware used by the Combined Compression Approach cannot be justified if it is compared with that of the previous approaches, because its total reduction in error masking is not better than that obtained in the previous approaches. More specifically, neglecting the small amount of error masking probability introduced by a parity tree ( $2^{-l}$ , where  $l$  is usually in the order of thousands or more), the total  $pr$  is the product of that of two final compressors. It was shown earlier in this section that the masking probability of a  $k$ -bit LFSR is  $2^{-k}$ ; whereas, in general, the probability for a counter of the same length is considerably more than that of the LFSR. Consequently, the total  $pr$  in this example is more than the usual  $2^{-2k}$ .

Other examples of this Combined Compression Approach suggest different combinations of compression techniques. However, the reduction in deception volume ob-

tained, in general, remains no better than in the previous example. One exception is the Accumulator Compressor approach [Saxena 85] developed for single output circuits. However, in this case, a much more considerable amount of additional hardware is required due to the adoption of an extra adder-accumulator, making the approach not very feasible.

Based on the analysis of the above mentioned approaches, it can be concluded that in general, the same reductions in error masking are obtained for the same amounts of extra hardware. A general theorem along these lines is proven in [Bhavsar 84]. This theorem states that the error masking probability of a  $k$ -bit compression function is  $2^{-k}$ , where  $k$  is the total number of bits in the final signature(s), if and only if the compression function is uniform. Since polynomial division-based compression functions are uniform (see section 3.4), and since all the above mentioned approaches use polynomial division-based compression functions, therefore, there exists no provision to obtain better than  $2^{-k}$  error masking probability under such approaches.

As a result the following problem is posed:

Can the probability of error masking be decreased to very small values, such as  $2^{-\text{thousands}}$ , without increasing  $k$  to thousands, i.e. keep extra hardware requirements small, and keeping the value of  $l$  not very large? Such a level of certainty is particularly suited for requirements of very high test quality, e.g. 1 failure in a  $10^5$  or  $10^6$  parts.

The work presented in this dissertation provides a positive solution to this problem. It will be shown that under the same assumptions as the described approaches the desired error masking probability ( $2^{-\text{thousands}}$ ) can be achieved by utilizing approximately the same amounts of additional hardware. The concept behind this solution is presented next.

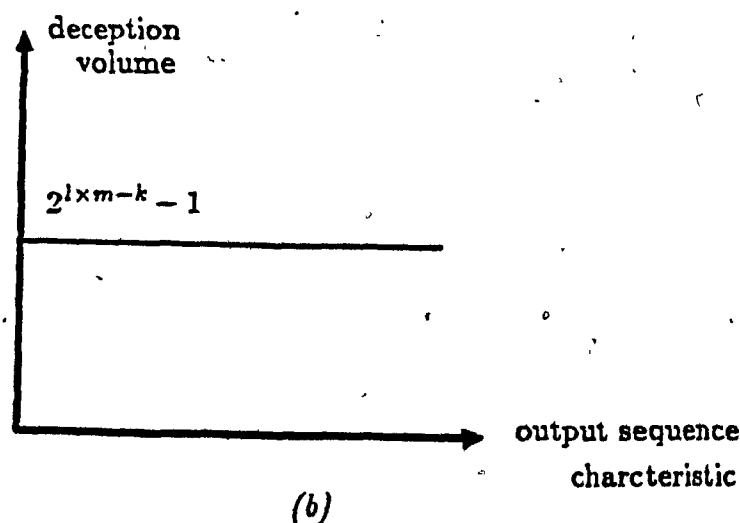
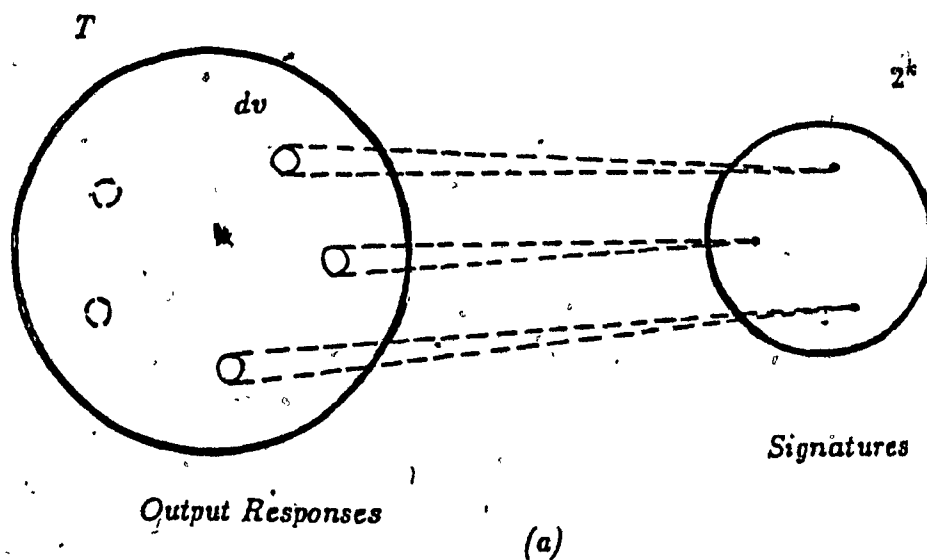


### 3.4 The ODM Concept:

To reduce the deception volume, and thus the error masking probability, all the previous approaches either developed a sophisticated compression technique or modified the existing techniques. However, it is important to realize that the error masking problem not only depends upon the output data compression technique adopted, but also on a second factor, viz. the error-free output data itself.

The superiority of the ODM scheme presented in this work results from the fact that the dependence on both factors is exploited [Agarwal 83], and adopted in a complete BIST scheme with a small area overhead realization [Zorian 84] [Zorian 86c]. The sophisticated compression function constituting the first factor is only considered in detail in the following chapters; whereas the second factor, i.e. the idea of utilizing the error-free output data, is discussed next.

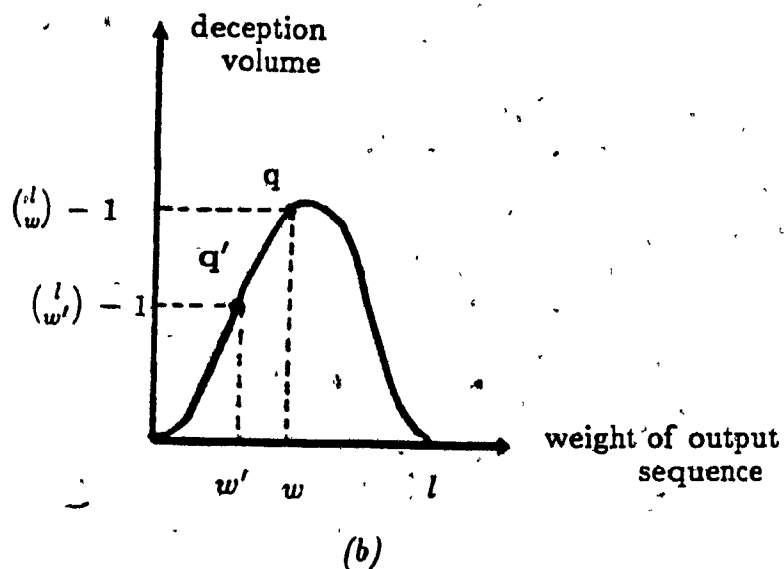
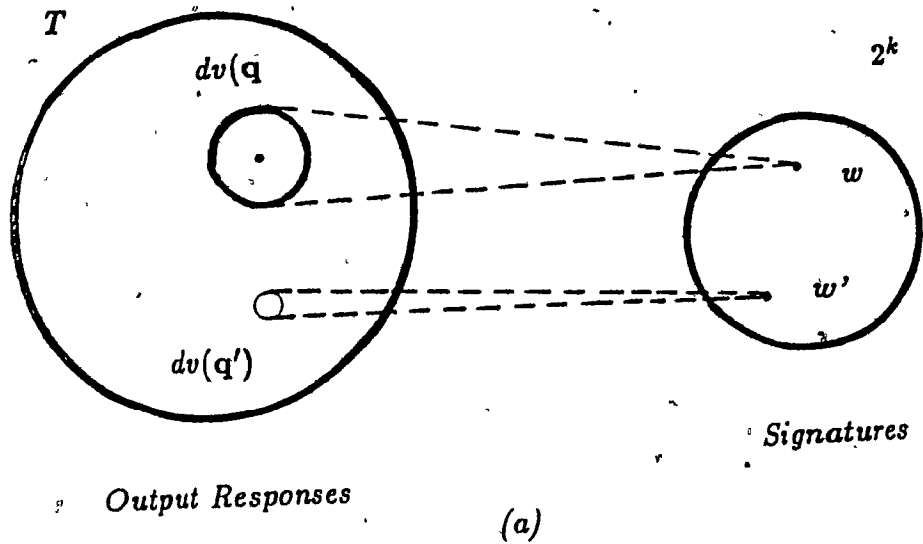
Under certain compression functions, the deception volumes are independent of the actual output data. In such cases, the compression function produces equal deception volumes for each of its input matrices ( $l \times m$ ). Hence, such a compression function is said to have a uniform deception volume. More formally, the deception volume is considered uniform if a compression function partitions the space of its input data matrices equally, as shown in Figure(3.8-a), where the space of input data matrices consists of all possible  $2^{l \times m}$  matrices. The famous polynomial division-based compression technique as well as the parity check-based technique are good examples of functions that have uniformly distributed deception volumes. Figure(3.8-b), where each point on the x-axis refers to a specific  $l \times m$  matrix, shows for an MISR a constant deception volume with respect to a characteristic (i.e. weight, transition count, etc..) of the output data matrices. Hence, all output data matrices mapped under such a compression function yield the same deception volume.



**Figure (3.8) Uniform Deception Volume (MISR)**

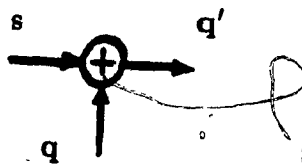
On the other hand, if a compression function partitions the same space of all possible input matrices into unequal regions, as shown in Figure(3.9-a), then its deception volume is non-uniform. In this case, the deception volume depends on the input data. The count-based compression techniques like One's Counting, Transition Counting and Edge Counting, have a non-uniform deception volume. Their respective deception volumes are a function of weight, number of transitions, and number of edges. For instance if One's Counting is applied on a single output CUT, then the resulting signature is the weight of its  $l$ -bit long output string. Since the weight of this string is generally close

to  $l/2$ , its deception volume, as shown in Figure(3.9-b), is  $\binom{l}{l/2}$ . However, if the string's weight is either close to zero or one, then the deception volume is much smaller.



**Figure (3.9) Non-Uniform Deception Volume (One's Counting)**

Any modification in output data may change the deception volume only if it is non-uniform. The purpose of the output data modification concept is precisely to take advantage of this non-uniform distribution, to reduce the deception volume of the output sequence. Figure(3.10) shows a reduction in deception volume realized by modifying the output sequence  $q$  by using a modifier sequence  $s$  resulting in a sequence  $q'$  which has a deception volume much smaller than that of  $q$ ,  $dv(q') \ll dv(q)$ . This modification operation, because it is performed by Exclusive-Or, results in no information loss. Hence, the original string  $q$  can be fully recovered from  $q'$  by performing the bit-wise Exclusive-OR with  $s$  again. However, this scheme is practical only if the modifier string  $s$  can be generated easily and by inexpensive hardware.



Figure(3.10) The Modification Operation

The ideal value for the modifier string  $s$  is the original string  $q$  itself, such that after the bit-wise Exclusive-OR is performed,  $q'$  is the string of all zeros, i.e.  $dv(q') = 0$ . Since this is a likely expensive possibility in reality,  $s$  has to be as close as possible to the original string  $q$  in the case of One's Counting-based compression. The closer the provided  $s$  is to  $q$ , the smaller is the deception volume obtained:

$$dv(q') = \binom{l}{w'} - 1 \ll dv(q) = \binom{l}{w} - 1$$

where  $w$  is the weight of the string  $q$ , and  $w'$  is the weight of  $q'$ .

This concept of modifying the output data to provide higher certainty of error coverage is adopted by a new BIST scheme called output data modification (ODM) scheme. The model of this scheme is presented next.

### 3.5 The Proposed ODM Model:

In order to design a BIST model which realizes the ODM scheme, the following considerations have been taken into account:

- Provide the design of a general BIST model which implements a large set of compression functions  $\{f_j\}$ , such that this set provides the possibilities of obtaining extremely high improvements in error coverage.
- Exploit the functionality of the given CUT to select a circuit-specific compression function  $f^*$  from the available set  $\{f_j\}$ , for which  $dv(A_0, f^*) < dv(A_0, f_i), \forall i \neq *$ .
- The silicon area used to realize the entire model (including the circuit-specific part) should not, in general, be much more than the improved BIST schemes mentioned in section 3.3.

The ODM scheme which is the realization of such a model will be described throughout the following chapters. However, the design of a general BIST model is demonstrated in the setup in Figure(3.11). This setup consists of a dotted box representing the standard form of BIST schemes, along with two additional blocks. The block on the left is called the Modifier. Its task is to generate the modifier sequence  $s$ , whereas the one on the right is the Improved Compressor.

The generic model of the ODM has the capability of providing a large set of compression functions, particularly since it contains a special functional block, the Modifier, which has a large choice of programmable functions. An appropriate function for this

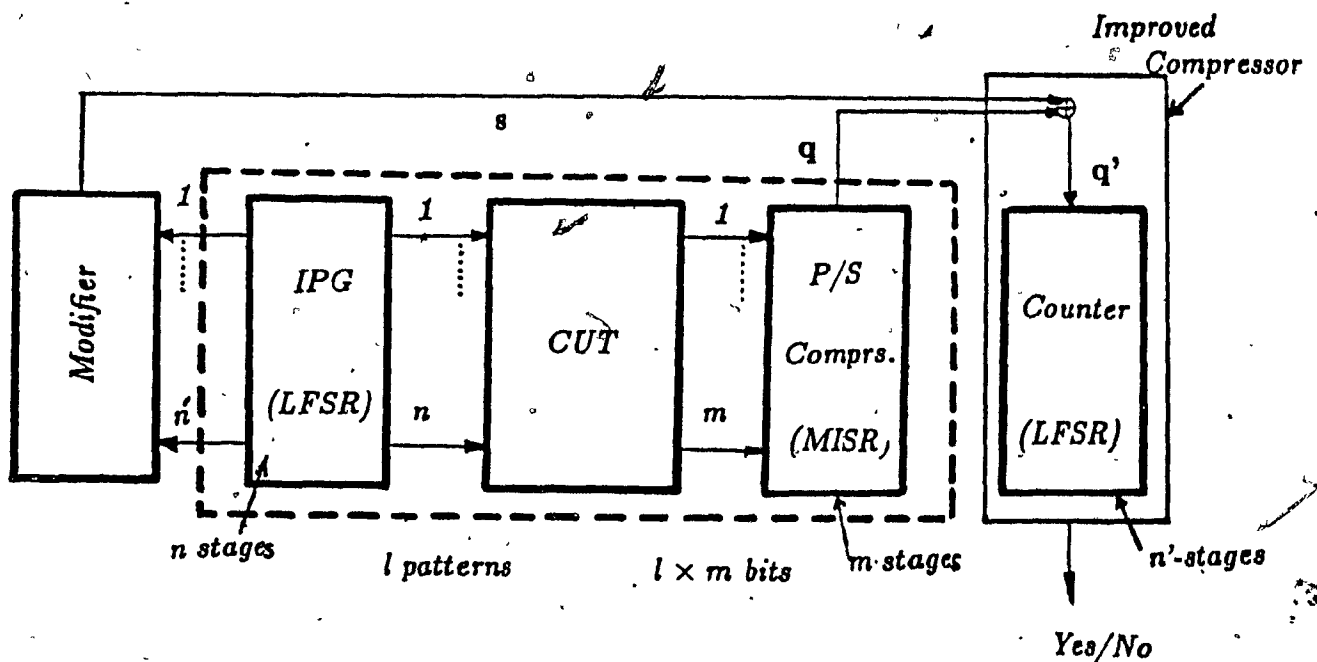


Figure (3.11) A BIST Model for the ODM Scheme

Modifier block is selected such that it generates the best  $s$  possible for the modification operation. Such selection has to be made according to the functionality of the CUT. Hence, the Modifier block necessitates a circuit-specific design with each CUT.

In the Improved Compressor block, the output data modification operation is performed. In the same block, the non-uniform output data compression is also performed. This block receives as input two  $l$ -bit long streams:  $s$  from the Modifier block, and the quotient bit stream  $q$  produced by the MISR. The two streams of data,  $s$  and  $q$ , are Exclusively-ORed to produce the modified  $q$  denoted by  $q'$ . Finally,  $q'$  is compressed by a count-based compression technique.

For instance, if One's Counting is adopted for compressing a modified sequence, then the deception volume is changed from  $\binom{l}{w} - 1$  to  $\binom{l}{w'} - 1$ , where the number

of 1's in  $q$  is  $w$  and that in  $q'$  is  $w'$ . Correspondingly, the error masking probability is reduced from  $((\binom{l}{w} - 1)/(2^l - 1))$  to  $((\binom{l}{w'} - 1)/(2^l - 1))$ . Assuming  $l = 4000$ ,  $w = 1900$  and  $w' = 700$ , the two masking probabilities are respectively  $2^{-21}$  and  $2^{-1400}$ .

A detailed description of both the the generic and circuit specific designs of the blocks comprising the ODM model, is presented in chapter 6. To generate an  $s$  as close possible to  $q$ , the important problem of generating a modifier sequence from a block requiring a very small silicon area must be solved. Several sequence generation methods are developed for this purpose in chapter 4.

In the remainder of this dissertation, it is assumed that the count-based compression in the Improved Compressor is the One's Counting technique. However, the discussions of the next chapters can easily be extended to other count-based techniques as well as other nonlinear compression methods [Saxena 85] besides counting.

## Chapter 4

### Modifier Sequence Generation Methods

#### 4.1 Introduction:

The problem of error masking in *BIST* is analyzed in the previous chapter, and a new *BIST* scheme based on output data modification is suggested as a solution to this problem. However, since the hardware constraint is one of the most important considerations in *BIST* schemes, this promising approach can not be realized if the cost of its hardware implementation is high. Therefore, it is crucial to provide an inexpensive realization of the *ODM* scheme.

The *Modifier* block is the only extra block designed particularly for the self-test structure of the *ODM* model (unlike the *IPG* and the *ODC*). Therefore, an important consideration, under the *ODM* scheme, is to provide a *Modifier* block which utilizes a small amount of extra hardware. However, this is not a simple task since this block



must generate a sequence  $s$  of length  $l$  (test length) which is as close as possible to the quotient sequence  $q$ , with a small amount of extra hardware. It is obvious that with no hardware constraints, the Modifier block can have a very simple realization. For example, it could simply be, the repetition of the original CUT. Clearly, this is a simple but expensive solution. The major task here is to design a block, with a small amount of extra hardware, which has the capability of generating a large set of sequences, so that the possibility of finding a sequence closer to the original string  $q$  is increased. Hence, several methods for generating such sequences are developed, along with their inexpensive hardware realizations and appropriate circuit-specific implementation algorithms.

In the following discussions of the sequence generation methods, we use the term easily generatable sequence (EGS) for an  $l$ -bit long sequence generated by  $O(n')$  gates, where  $n' = \lfloor \log(l) \rfloor$ . In fact, the different sequence generation methods described are intended to generate EGS's, and hence use only  $O(n')$  gates. These generation methods are classified under two types: methods which generate a sequence by segments [Agarwal 83] [Zorian 84], considered in section 4.2; and a method which generates a sequence by totality [Zorian 86a], described in section 4.3.

## 4.2 Methods to Generate a Sequence by Segments:

A sequence generation method belongs to this type if it produces an EGS by generating its segments and concatenating them using combinational circuits. Several methods of this type, the exponentially increasing segments (EIS) methods, and the equal size segments (ESS) method, are developed for modifier string generation, and are introduced below.

#### 4.2.1 EIS Methods for Modifier Sequence Generation:

Several EIS methods have been developed to generate sequences composed of segments that are of exponentially increasing lengths. The simplest method developed to generate such a sequence is based on one seed and requires a single input [Agarwal 83]. An extension of this method to one with multiple choices of seeds increases the number of possible EGSs [Zorian 84]. Moreover, the development of another method which uses double inputs, provides a major improvement in the number of possible EGSs [Zorian 84]. All three methods and some further EIS possibilities are discussed in the following.

##### 4.2.1.1 EIS with Single Input and a Single Seed:

A simple method to generate sequences with exponentially increasing segments is the single input and single seed method. Although a more advanced double seed method which uses the same hardware structure and that provides a much higher number of EGSs will be presented in section 4.2.1.2, the single seed method, developed in [Agarwal 83], is also described here for the sake of completeness.

A sequence  $s^i$  of length  $2^i$  is said to consist of exponentially increasing segments if  $s^i$  is recursively obtained by concatenating two subsequences of identical length,  $2^{i-1}$ . Each sequence is generated by starting with a subsequence  $s^0$  of length 1, and recursively building subsequences of exponentially increasing lengths. More specifically,  $s^1$  consists of  $s^0$  as one of its halves, and a sequence  $r^0$  of length 1 as the other half, where  $r^0$  is  $s^0$ ,  $\bar{s}^0$ ,  $Z^0$  or  $1^0$ , and

$$s^1 = r^0.s^0$$

Here  $Z^i$  and  $1^i$ ,  $0 \leq i \leq n'$ , refer to a sequence of length  $2^i$  consisting of all zeros and all 1's respectively. For convenience, we assume that  $l = 2^{n'}$ , where  $n' \leq n$ . Thus the

more general form can be given as

$$s^{i+1} = r^i \cdot s^i$$

where  $r^i = s^i, \bar{s}^i, Z^i$  or  $1^i$ ,  $0 < i < n' - 1$ .

Recall that an  $s^{n'}$  of length  $l$  as close as possible to a given  $q$  is to be generated in the *Modifier* block. Thus, the generation of  $s^{n'}$  is governed by the makeup of  $q$ . In other words, at each stage  $i$  of generation, we make the choice of  $r^i$  from the four possible options based on what the best choice for a given  $q$  is.

**Lemma 4.1:** The total number of EGSs of length  $l = 2^{n'}$  is  $O(2^{2n'})$ , and thus their ratio to the total number of sequences of length  $l$  tends to 0 for large  $l$ , that is,

$$\frac{2^{2n'}}{2^l} \rightarrow 0 \text{ for } l \rightarrow \infty$$

**Proof:** Let  $N(n')$  denote the total number of EGSs of length  $2^{n'}$ . Then from the above definition of EGS, we have  $N(n') \simeq 4N(n' - 1)$ . A simple iteration shows that

$$N(n' + 1) \simeq 4^{n'+1} N(0) \simeq 4^{n'+1} \cdot 2 \simeq 2^{2n'+3}$$

Thus  $N(n') = O(2^{2n'})$ .

Given a sequence  $q$ , we then have to determine an optimal EGS such that the weight of  $q \oplus s$  is minimal over all possible choices of  $O(2^{2n'})$  sequences. Due to the exponential size of the choices to be made, an exhaustive search can become very expensive. In a typical application,  $l$  may be equal to  $2^{20}$  where 20 is the number of input lines to the CUT. The total number of EGS's then would be  $2^{40} \simeq 10^{12}$ . In the following, we thus propose a heuristic algorithm of  $O(n')$  to determine an  $s$  for a given sequence  $q$  of length  $l = 2^{n'}$ .

**Algorithm (1):**

```

begin
  let  $s^0$ , the most-significant-bit of  $s$ , be the same as  $q^0$ ,
    which is the most-significant bit of  $q$ ;
  for  $i = 0$  to  $n' - 1$  do
    begin
      let  $s^i$  be the segment generated so far;
      let  $q^i$  be  $s^i$ 's corresponding segment on  $q$ ;
      select  $r^i$  among  $\{s^i, \overline{s^i}, Z^i, 1^i\}$  such that:
         $r^i \oplus q_i^i$  has minimal weight
      (where  $q_i^i$  represents the  $2^i$ -bit long segment to the left of  $q^i$ );
       $s^{i+1} := r^i \cdot s^i$ 
    end;
   $s := s^{n'}$ 
end.

```

**Example 4.1:** Let  $q$  be as given below:

$q =$	1010-0100 1001 0010	
$s^0 =$	0	$r^0 = \overline{s^0}$
$s^1 =$	10	$r^1 = Z^1$
$s^2 =$	0010	$r^2 = \overline{s^2}$
$s^3 =$	1101 0010	$r^3 = Z^3$
$s^4 =$	0000 0000 1101 0010	
$q' =$	1010 0100 0100 0010	

Clearly the weight of  $q$  is  $w = 6$ , whereas that of  $q'$  becomes  $w' = 4$ . Notice that in choosing  $r^3$ , there were two options; the other being 0010 1101. There is no restriction in making a choice in such situations.

**Lemma 4.2:** The number of bit-operations in Algorithm (1), is  $O(l)$  and the number of bit-vector operations is  $O(\log(l))$ , where  $l = 2^{n'}$ .

**Proof:** For  $i = 0$  to  $n' - 1$ , we notice that a constant number of bit-vector operations are performed to determine each  $s^i$ . However, since each bit-vector operation corresponds to an increasing number of such bit-operations, the total number is  $O(2^{n'})$ .

**Lemma 4.3:** The weight  $w'$  of  $q' = q \oplus s$  is always less than or equal to  $w$ , wherein  $w$  is the weight of  $q$ , and  $s$  is obtained from Algorithm (1).

**Proof:** Note that, at every iteration step for  $i = 0, 1 \dots n' - 1$  in Algorithm (1), there are four choices one of which is  $Z^i$ . Therefore, in the worst case, when  $s^i$  cannot decrease the weight of  $q^i$  at all, then the choice for  $s^i$  is  $Z^i$ . Therefore  $w' \leq w$ .

Suppose now that  $w - w' = u$  is the reduction in the weight of  $q$ . This reduction will lead to a very large reduction in the modified sequence. More specifically, let  $u/w = d$  and  $l/w = b$ . If we assume that  $d \geq .07$ , then it is possible to prove the following:

**Theorem 4.1:** The deception volume of  $q'$  is at least  $2^{1u}$  times smaller than the deception volume of  $q$ . In other words:

$$dv(q) \geq 2^{1u} \cdot dv(q')$$

**Proof:** The proof is largely combinatorial in nature and relies on Sterling's formula for factorial approximation. First of all, we have that

$$\begin{aligned} dv(q) &= \binom{l}{w} = \frac{l!}{w!(l-w)!} \\ &= \left(\frac{l}{e}\right)^l \sqrt{2\pi l} \frac{1}{\left(\frac{w}{e}\right)^w \sqrt{2\pi w} \left(\frac{l-w}{e}\right)^{l-w} \sqrt{2\pi(l-w)}} \\ &= \frac{l^l}{w^w (l-w)^{l-w}} \sqrt{\frac{2\pi l}{2\pi w 2\pi(l-w)}} \dots \end{aligned} \quad (4.1)$$

Similarly

$$\begin{aligned} dv(q') &= \binom{l}{w'} = \binom{l}{w-u} \\ &= \frac{l^l}{(w-u)^{w-u} (l-(w-u))^{l-(w-u)}} \sqrt{\frac{2\pi l}{2\pi(w-u) 2\pi(l-(w-u))}} \end{aligned} \quad (4.2)$$

Notice that in equation(4.2) if  $w - u$  is very small, then the use of the Sterling's formula will be invalidated. Keeping that restriction in mind, we next calculate the ratio

$$\begin{aligned} \frac{dv(q)}{dv(q')} &= \frac{w - u^{w-u} (l - (w - u))^{l-(w-u)}}{w^w (l - w)^{l-w}} \sqrt{\frac{(w - u) (l - (w - u))}{w (l - w)}} \\ &= \left(1 + \frac{u}{l - w}\right)^{l-w} \cdot \left(1 - \frac{u}{w}\right)^w \cdot \left(\frac{l}{w - u} - 1\right)^u \cdot R \end{aligned} \quad (4.3)$$

wherein  $R$  is the square root expression. Substituting now  $u = dw$  and  $l = bw$  and rearranging the terms in equation (4.3) we get

$$= \left( \frac{(b - 1 + d)^{b-1+d} (1 - d)^{1-d} w}{(b - 1)^{b-1}} \right) \cdot R$$

If we now take the  $\log_2$  on each side, it follows that

$$\begin{aligned} \log_2 \frac{dv(q)}{dv(q')} &= w((b - 1 + d)\log_2(b - 1 + d) + (1 - d)\log_2(1 - d) \\ &\quad - (b - 1)\log_2(b - 1)) + \log_2 R \end{aligned} \quad (4.4)$$

It can be easily shown that expression (4.4) is an increasing function for increasing values of  $d$  and  $b$ . For the minimum value of  $b = 2.0$  (since  $w \leq l/2$ ), we have that

$$\log_2 \frac{dv(q)}{dv(q')} = w((1 + d)\log_2(1 - d) + (1 - d)\log_2(1 - d)) + \log_2 R \quad (4.5)$$

where  $R = \sqrt{(1 - d^2)}$ . Thus,  $\log_2 R = 1/2 \log_2(1 + d) + 1/2 \log_2(1 - d)$ . Since,  $w$  is bound to be a very large quantity in BIST, the influence of  $\log R$  in equation (4.5) can be ignored. Finally, it can be shown that for  $d \geq .07$ ,  $(1 + d)\log(1 - d) + (1 - d)\log(1 - d) \geq .1d$ . Therefore, it follows that

$$\log_2 \frac{dv(q)}{dv(q')} \geq .1dw = .1u$$

or

$$dv(q) \geq 2^{.1u} \cdot dv(q')$$

**Example 4.2:** Consider a multiplier circuit for any 8 bit by 8 bit multiplications of positive integers. If we now apply  $l = 2^{16}$  input vectors to the circuit, the weight  $w$  of the most significant bit sequence  $q$  is calculated to be 9918. Therefore,

$$dv(q) = \binom{65536}{9918}$$

and

$$pr = \frac{\binom{65536}{9918}}{2^{65536}}$$

However, if Algorithm(1) is used to determine a modifier sequence and  $q$  is modified to  $q'$  then the weight  $w'$  of  $q'$  is 4036, leading to  $u = w - w' = 5612$ . Therefore

$$dv(q') \leq 2^{-.1u} dv(q) = 2^{-561} dv(q)$$

and similarly

$$pr' \leq 2^{-561} pr$$

Since only  $O(n') = O(16)$  gates were used, the reduction of  $2^{-561}$  is very significant. In fact, since  $d = u/w = .566$  and  $b = l/w = 6.60$ , if we use equation(4.4), we get that

$$\begin{aligned} \log \frac{dv(q)}{dv(q')} &= 9918(6.17 \log(6.17) + .434 \log(.434) \\ &\quad - 5.60 \log(5.60) + \frac{1}{2} \log(.48)) \simeq 17262 \end{aligned}$$

Therefore, to be exact  $dv(q') = 2^{-17262} dv(q)$ .

It follows from Example(4.2) that Theorem(4.1) provides only a lower bound on the reduction. For  $d$  significantly greater than .07, the reduction is bound to be much higher than  $2^{-.1u}$ .

#### 4.2.1.2 EIS with Single Input and Multi-Seed Possibility:

It is possible to extend the single input and single seed method to reach larger choices of EGS's by selecting the very first bit  $s^0$  from any specific bit position out of the  $l$  possible positions of  $q$  [Zorian 84]. We refer to such a position and any given value (0 or 1) for that position as a seed. Starting from a seed  $s^0$ , we start building  $s^i$ ,  $i = 1, \dots, n'$ , by choosing an  $r^{i-1}$  which is either to the left or to the right of  $s^{i-1}$ . In a general form, this can be written as:

$$s^i = s^{i-1} \cdot r^{i-1}$$

or

$$s^i = r^{i-1} \cdot s^{i-1}$$

The multi-seed possibility is demonstrated in the following example.

**Example 4.3:** Let  $q$  be as given below and the seed be the thirteenth bit from the left.

$q = 1101\ 0011\ 1000\ 0101\ 1001\ 0110\ 1100\ 0001$	
$s^0 = 0$	$r^0 = \overline{s^0}$
$s^1 = 01$	$r^1 = s^1$
$s^2 = 0101$	$r^2 = \overline{Z^2}$
$s^3 = 0000\ 0101$	$r^3 = \overline{s^3}$
$s^4 = 1111\ 1010\ 0000\ 0101$	$r^4 = s^4$
$s^5 = 1111\ 1010\ 0000\ 0101\ 1111\ 1010\ 0000\ 0101$	
$q' = 0010\ 1001\ 1000\ 0000\ 0110\ 1100\ 1100\ 0100$	

$$w = 15 \quad w' = 11$$

Since generating EGS's with EIS methods has an exponentially increasing nature, we need  $n'$  generation stages to obtain  $s$  of length  $l$ . A hardware realization of such a Modifier block is shown in Figure(4.1), wherein each generation stage  $i$  ( $i = 1, 2, \dots, n'$ ) is represented by a functional box  $G^{i+1}$ , realized by one simple two-input gate. The first input provides the previously generated subsequence  $s^i$ , and the second provides a clock input  $c^i$  which is the  $i$ -th output of an  $n'$ -bit counter. If a



counter of length  $n' = n$  is needed for exhaustive input pattern generation, this same counter can also be used to provide the  $c^i$  inputs required for each gate  $G^{i+1}$ , as seen in Figure(4.1). This eliminates the necessity of an extra counter in the Modifier block. Moreover, Table(4.1) helps to adopt the simple function  $g^{i+1}$  realized by gate  $G^{i+1}$  corresponding to every stage  $i$ . The total number of EGS's of length  $2^{n'}$  obtained by this scheme with multi-seeds can be shown to be  $O(6^{n'})$  by using this approach. The formal algorithm for this scheme is presented in Algorithm (2).

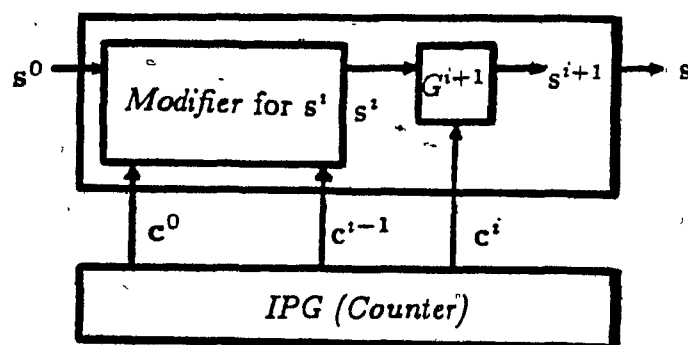


Figure (4.1) Single Input Method

Table (4.1)

Modifier Sequence Generation Functions

$s^{i+1}$	$g^{i+1}(s^i, c^i)$
$s^i.s^i$	$s^i$
$\overline{s^i}.s^i$	$s^i \text{ XOR } c^i$
$1^i.s^i$	$s^i \text{ OR } c^i$
$s^i.\overline{s^i}$	$\overline{s^i \text{ XOR } c^i}$
$s^i.Z^i$	$s^i \text{ AND } c^i$
$s^i.1^i$	$s^i \text{ OR } \overline{c^i}$

**Algorithm 2:**

```

begin
for each selected seed  $q^0$  do
begin
let  $s^0$  be equal to  $q^0$ ;
for  $i = 0$  to  $n' - 1$  do
begin
let  $s^i$  be the segment generated so far;
let  $q^i$  be  $s^i$ 's corresponding segment on  $q$ ;
select  $r^i$  among  $\{s^i, \bar{s}^i, Z^i, 1^i\}$  such that:
 $r^i \oplus q_h^i$  has minimal weight
(where  $q_h^i$  represents the symmetric half of  $q^i$ );
if ( $r^i$  is the right-half of  $s^{i+1}$ )
then  $s^{i+1} := s^i.r^i$ 
else  $s^{i+1} := r^i.s^i$ 
end;
let  $s^{n'}$  be the sequence which provides minimal  $s^{n'} \oplus q$ 
end;
 $s := s^{n'}$ 
end.

```

Note that any bit in  $q$  can be a seed. However, it is unrealistic to generate  $s$ 's based on every single seed and then select the best one. The problem of a priori selecting an efficient seed thus needs to be solved. Nonetheless, it may be noticed here that the number of possible EIS's is highest when the previously generated sequence  $s^i$  consists of alternative 0's or 1's, instead of containing a string of all 0's or all 1's, (for instance, if the sequence  $s^i$  consists of all 0's then  $s^{i+1}$  could also be all 0's, resulting in a low number of possible EGS's). Thus, the subsequences in  $q$ , that should be first considered to choose a seed from, are those where a number of transitions are concentrated.

**4.2.1.3 EIS with Double Inputs:**

In this method we provide the capability of generating even higher number of EGS's by adding a second (auxiliary) input sequence  $p^i$  for each generating stage  $i$ , where

$i = 1, \dots, n'$ , thus providing two distinct sequences  $s^i$  and  $p^i$  as bases to generate  $s^{i+1}$  and  $p^{i+1}$  [Zorian 84]. However, the additional sequence  $p^i$  necessitates an auxiliary line of gates ( $G^1, G^2, \dots, G^{n'}$ ) besides the main ones ( $F^1, F^2, \dots, F^{n'}$ ), as shown in Figure(4.2).

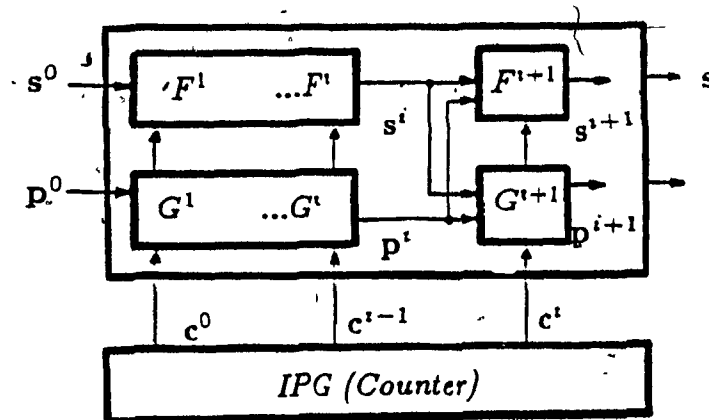


Figure (4.2) Double Input Method

The advantage of having two distinct sequences is that it provides the possibility of obtaining up to 16 (i.e.  $2^{2^2}$ ) choices of functions at each stage. Notice that, in contrast, in the previous method only four (i.e.  $2^{2^1}$ ) functions were possible. This causes an improvement in the total number of EGS's, thus increasing the availability of an  $s$  closer to the original  $q$ .

Each sequence  $s^{i+1}$  consists of two subsequences of identical length:

- one of them is the exponentially generated subsequence  $s^i$ ;
- the other is the auxiliary subsequence  $p^i$ , which itself consists of two independent halves:  $p_l^{i-1}$  and  $p_r^{i-1}$ . These two halves are generated based on both previous subsequences  $s^{i-1}$  and  $p^{i-1}$  by using the functions  $g_l^{i-1}$  and  $g_r^{i-1}$ , as stated below:

$$p_l^{i-1} = g_l^{i-1}(s^{i-1}, p^{i-1})$$

$$p_r^{i-1} = g_r^{i-1}(s^{i-1}, p^{i-1})$$

To determine the auxiliary functions  $g_l^{i-1}$  and  $g_r^{i-1}$ , and therefore, to determine the structure of  $G^i$ , we have to use the two subsequences  $s^{i-1}$  and  $p^{i-1}$ , for each one of the 16 obtainable functions, i.e. the 16 distinct half  $p$ 's ( $p_l^{i-1}$ ,  $p_r^{i-1}$ ). For each half  $p^i$ , we have to select the closest possible choice to its corresponding part on the original sequence  $q$ . These steps are shown in Algorithm(3) and applied on Example(4.4).

**Algorithm 3:**

```

begin
  for each selected seed  $q^0$  do
    begin
      let  $s^0$  be equal to  $q^0$ ;
      let  $p^0$  be the symmetric bit to  $q^0$  on  $q$ ;
      for  $i = 0$  to  $n' - 1$  do
        begin
          let  $s^i$  be the segment generated so far;
          let  $s^{i-1}$  and  $p^{i-1}$  be the two segments of  $s^i$ ;
          let  $q^i$  be  $s^i$ 's corresponding segment on  $q$ ;
          select the function  $g_l^{i-1}$  among the 16 possible functions;
           $p_l^{i-1} = g_l^{i-1}(s^{i-1}, p^{i-1})$ 
          such that:  $p_l^{i-1} \oplus q_l^{i-1}$  has minimal weight;
          select the function  $g_r^{i-1}$  among the 16 possible functions;
           $p_r^{i-1} = g_r^{i-1}(s^{i-1}, p^{i-1})$ 
          such that:  $p_r^{i-1} \oplus q_r^{i-1}$  has minimal weight;
           $p^i := p_l^{i-1} \cdot p_r^{i-1}$ ;
          if ( $p^i$  is the right-half of  $s^{i+1}$ )
            then  $s^{i+1} := s^i \cdot p^i$ 
            else  $s^{i+1} := p^i \cdot s^i$ 
          end;
        end
      let  $s^{n'}$  be the sequence which provides minimal  $s^{n'} \oplus q$ 
      end;
    end;
  end.

```

**Example 4.4:** The Double Input Method is applied on the same  $q$  as in Example(4.3):

$q =$	1101 0011 1000 0101 1001 0110 1100 0001	
$s^0 =$	0	$p^0 = \overline{s^0}$
$p^0 =$	1	
$s^1 =$	01	$p^1 = s^0 \cdot s^0$
$p^1 =$	00	
$s^2 =$	0001	$p^2 = \overline{p^1} \cdot p^1$
$p^2 =$	1100	
$s^3 =$	1100 0001	$p^3 = (p^2 + s^2) \cdot (\overline{p^2 + s^2})$
$p^3 =$	1101 0110	
$s^4 =$	1101 0010 1100 0001	$p^4 = (p^3 + s^3) \cdot s^3$
$p^4 =$	1101 0011 1100 0001	
$s^5 =$	1101 0011 1100 0001 1101 0010 1100 0001	
$q' =$	0000 0000 0100 0100 0100 0100 0000 0000	

$w=15 \quad w'=4$

The use of two subsequences in each stage results in a higher number of obtainable functions. The total number of EGS's in the double input method is  $O(256^{n'})$ . If the number of input subsequences entering each generation stage  $i$  is increased, then one can achieve an even higher number of generatable sequences:  $2^{2^{(\# \text{ of input sequences})}}$ . However, in that case, the complexity of Algorithm(3) will be increased in the same order. This is because its complexity depends on the number of generatable sequences as it tries every possible function and then selects the appropriate one. Moreover, the silicon area requirement for the generation block is bound to increase as well.

#### 4.2.1.4 Further EIS Possibilities:

The modifier sequence obtained through the previous methods may not always result in drastic reductions in deception volume. In fact, no single method can always be suitable for all sequences of interest. Thus, to enhance the scope of modifier sequences, further techniques to enhance the EIS generation methods will be considered.

A particularly interesting technique, in this regard, is the simple Transition Counting technique. Transition Counting, in fact, counts the 0-to-1 and 1-to-0 transitions. This is equivalent to the weight of the sequence  $c$ , where the  $x$ -th bit of  $c$  is:  $c(x) = q(x) \oplus q(x-1)$ , and  $c(0) = q(0)$ , as shown in the following example.

**Example 4.5:** For instance, if

$$\begin{aligned} q &= 01011010 \\ \text{Shifted-}q &= 01011010 \\ c &= 011101110 \end{aligned}$$

then it can be noticed that the weight of  $q$  is 4; whereas the weight of the new sequence  $c$  is 6. If  $c$  is treated as the modified  $q$  itself, then the One's Counting applied on it results in:

$$dv(c) = \binom{8}{6} = 28 < dv(q) = \binom{8}{4} = 70$$

Notice that  $c$  obtained in this way always has one extra bit than  $q$ . However, without any loss of generality we can ignore the most significant bit. Interestingly,  $c$  has the recovery property in that  $q$  can be obtained from  $c$  using the following rules:

$$q(0) = c(0)$$

$$q(x) = c(x) \oplus q(x-1)$$

Thus,  $c$  has the same fault coverage information as  $q$  does.

Another possibility to enhance the scope of modifier sequences is to use a two level modification technique [Agarwal83]. For instance,  $c$  can be treated further by Algorithm(3) to determine a modifier string  $s$  for  $c$  such that the resulting  $c' = c \oplus s$  has lesser weight than  $c$ . This technique is shown in Figure(4.3). In example (4.5),  $c$  is so regular that an  $s = 11101110$  can be generated by Algorithm(3) leading to  $c'$  consisting of all zeros. The deception volume of  $c'$ , in that case, is of course 0.

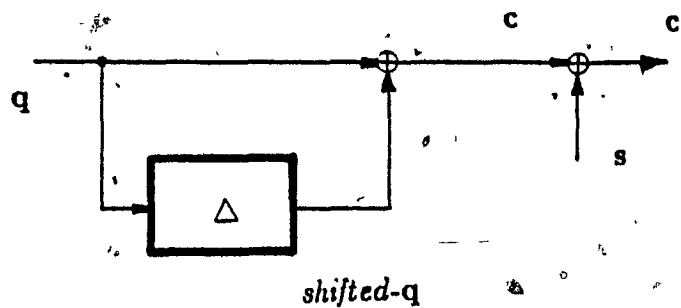


Figure (4.3) Two Level Modification Technique

One can easily increase the set of EGSs by employing some rather simple techniques. The above two-level modification technique is a good example of this. The  $c$  obtained from  $q$  and shifted- $q$  in fact becomes an EGS in the sense that it can be easily generated with  $O(n')$  hardware.

Another set of sequences that may be used in the sequence generation process is the set of strings which are already available on the BIST chip as a part of some other subsystem. In general, we will refer to such sequences as available sequences. A combination of EGSs and available sequences can be used to generate modifier sequences which do not belong to either sets. In fact, a method called ESS [Zorian 84], described in the following section, is totally based on the available sequences provided by the IPG.

#### 4.2.2 ESS Method for Modifier Sequence Generation:

One major difficulty with the EIS sequence generation methods, in general, is that for each generating stage  $i$ , a gate  $G^i$  is used to produce a segment of  $s$  of the same length as the cumulative length of the previously generated segments produced by the gates  $(G^1, G^2, \dots, G^{i-1})$ . This leads to lesser control on how close  $s$  can be generated

to a given  $q$ . The equal size segments (ESS) method described here overcomes this difficulty.

The ESS method generates a sequence of length  $l$  consisting of  $n' = \log(l)$  equal size segments, each of length  $l/n'$ . Hence, a subsequence  $s_i$ ,  $1 \leq i \leq n'$ , must be generated for each of the  $n'$  segments. Each  $s_i$  is generated using some available sequences on the chip, and a combinational gate  $G^i$  is designed for that specific  $s_i$ , as shown in Figure(4.4). The available sequences are assumed to be from the outputs of the IPG.

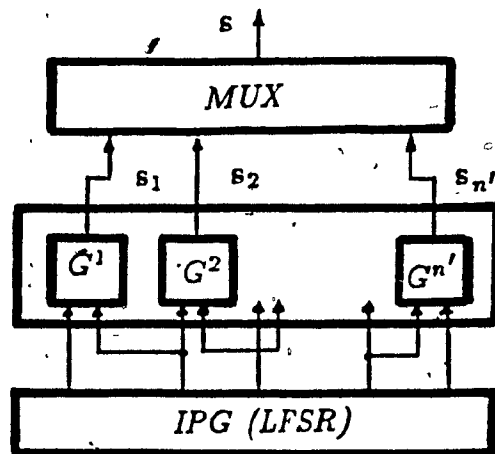


Figure (4.4) Equal Size Segments Method

Generally, the larger the number of input sequences available at each generating stage, the larger the number of EGS's that can be generated are. This can be noticed in the above Double Input Method when it is compared to the Single Input Method. If, for instance, four input sequences are used to generate each segment, then we can obtain up to  $2^{2^4}$  subsequences from that segment alone. Importantly, under the ESS method, there is the flexibility to use a different number of input (available) sequences at each stage. This means that not all the segments need to be generated with the same number of input sequences. Hence, in practice, the number of input sequences fed



to each stage is determined by considering the segment to be generated, as well as the available sequences on the chip. As the  $s_i$ 's are being generated by their respective gates, they are concatenated by a multiplexor to finally provide the required modifier sequence  $s$ . Because of its flexibility to use a different number of input sequences, the ESS method provides the possibility of generating a much larger number of EGSs than the previous methods. However, under ESS, the total number of possible EGSs cannot be predicted beforehand, since this number depends upon the number of available sequences at each stage.

Various limitations concerning the above methods which generate sequences by segments are discussed in the following section.

#### 4.3 A Method to Generate Modifier Sequences by Totality:

The important problem of generating large number of sequences with small amounts of hardware was the focus of the previous section. There, different methods which generate sequences by segments were presented, along with general designs to realize each method, some heuristic algorithms to implement them given a sequence  $q$ , and discussions about the capabilities of each method to generate a large number of EGSs. It is important to indicate that these methods, despite their capability for generating large number of sequences from small amounts of hardware, are not adequate if one wants to reduce the deception volume to any predetermined value. Furthermore, these methods are not guaranteed to work effectively for all cases. This is because they are based on local optimization (since any sequence provided by such a method is composed of optimal segments for each generating stage  $i$ ), and it is well-known that local optimization-based solutions are not necessarily globally optimal.

We overcome the above mentioned limitations by proposing a new method of sequence generation. This method generates a sequence  $s$  by totality [Zorian 86a], and provides a global optimization as well as a capability of reducing the deception volume to predetermined values. The basic idea behind this method is described in this section, along with some implementational aspects. However, the analytic proof which shows that this method provides a tremendous reduction in deception volume for any average circuit, is presented in chapter 5. The hardware realization of the Modifier block for this method is described in section 6.1.2. Finally, the implementation of the ODM scheme, using this method, is shown in section 6.2, along with procedures, heuristic algorithms, and examples.

A convenient way to describe the basic idea behind the method of generating sequences by totality is to think of the Modifier block as an  $n' = \log(l)$  input and single output combinational circuit, which produces  $l$  bits on the output line when  $l$  input patterns are applied to its inputs. In particular, if the  $l$  patterns are applied in a certain order, then the resulting  $l$ -bits form the modifier string  $s$ . Since the Modifier block realizes a single output function which generates a sequence in totality, a very convenient implementation for it is the functional block model used in [Mead 80]. This functional block design provides a large number of possible functions. However, given a CUT, a circuit-specific hardware programming is required in order to generate an  $s$  as close as possible to the quotient sequence  $q$ . In order to decide the circuit-specific functionality of the Modifier block, the method requires the function of the quotient sequence  $F_q$ . Now to find  $F_q$ , which we assume to have  $n'$  input variables, suppose that the  $l$ -bit long string  $q$  produced by the MISR corresponds to the output column of the truth-table of  $F_q$ . Then, we let all the input combinations of these  $n'$  input variables which become the input combinations to the Modifier block as well, be the input patterns generated

by the IPG, as Figure(3.11) illustrates. From the resulting truth table, it is possible to write a minimal sum-of-products form expression for the function  $F_q$  [Brayton 84] [Dagenais 85]. The functionality of the Modifier block, which is expected to be as close as possible to  $F_q$ , will then be obtained from the above mentioned sum-of-products expression.

Let  $t$  be the number of product terms in this sum-of-products expression  $F_q$ . Since the Modifier block is part of the area overhead needed to realize the ODM scheme, the amount of hardware to implement it should be as small as possible. Consequently, only a limited number of product terms, say  $p$ , might be allowed to be implemented. Clearly, if  $t \leq p$ , then all the product terms can be included in the Modifier block, and the modifier function is then the same as the desired function  $F_q$ . In that case the exact sequence  $q$  can be generated by the Modifier block, resulting in a  $q'$  of weight 0. However, if  $t > p$ , then the desired function  $F_q$  is not entirely realized, and a subset consisting of  $p$  product terms out of  $t$  have to be selected to form the modifier function which generates  $s$ . For this selection to be good, it should cover the largest number of 1-vertices in  $F_q$ , such that the selected product terms provide a global modifier function as close as possible to the original  $F_q$ . Such a selection algorithm is presented in section 6.2.2.

Although the global modifier function, under this method, depends upon the general function of the CUT, it is important to notice that it is not a repetition of this function.  $F_q$  is a much simpler function than that of the general CUT because the Modifier is an  $n'$ -input single-output function; whereas the CUT is an  $n$ -input and  $m$ -output function. Furthermore, the Modifier is stimulated by only  $l$  input patterns and not the set of all  $2^n$  possible patterns, where  $l \ll 2^n$  since pseudo-random testing is

considered. Moreover, since the functionality of a Modifier block is usually not identical to  $F_q$ , but as close as possible to it, the modifier function is, in general, a simplified approximation of it, realized by only a subset of  $F_q$ 's product terms.

The total number EGSs that can possibly be generated by a Modifier block under this method is flexible since it depends on the number of product terms  $p$  selected by the designer. Thus, the number of possible EGSs can be increased to any desired value by increasing  $p$ . This flexibility, as will be detailed in section 5.6, creates a trade-off between the area overhead required to implement the Modifier block and the number of generatable sequences. This is considered one of the important advantages of this method. Moreover, this method is generally very useful, since it can be proven that for a given  $p$ , it results in a significant reduction in deception volume for any average function created by sequence  $q$ . The formal proof of this statement is provided in chapter 5.

Because of the above mentioned advantages, the method to generate a sequence by totality is adopted to design and implement the Modifier block described in the remainder of this dissertation. The details about the general design of the Modifier block, under this method, and illustrations about determining its circuit-specific functionality are provided in chapter 6, along with the complete implementation of the ODM scheme.

## Chapter 5

### Reduction in Deception Volume for Average Sequences

The objective of this chapter is to show that in general, the deception volume of an average sequence of any specific weight is reduced by a significant amount by applying output data modification and that this reduction can be controlled by  $p$  (the predetermined number of product terms composing the Modifier block). In order to achieve this objective, first, the average number of product terms of a certain size, and the fixed number of isolated vertices is determined first. Then based on this determination, the weight of these product terms is derived, and then used to find out the reductions in weight and deception volume. The first section deals with certain preliminary calculations required to determine the average number of product terms.

#### 5.1 Preliminary Calculations:

Let a product term in an  $n'$  variable function be termed as a cube of size  $k$  (also denoted  $k$ -cube) if it covers  $2^k$  vertices in this function, where  $(k = 0, 1, \dots, n')$ . Such a

cube is represented when necessary by cubic notations [Dietmeyer 79]. For example, the product term  $a\bar{b}$  in a six variable function (i.e.  $n'=6$ ) covers  $2^4$  vertices. Therefore, it is considered a 4-cube, and its cubic notation is 10xxxx. In other respects, a sequence, in this chapter, is said to be composed of a set of  $k$ -cubes, since it is assumed to be generated by the (modifier) function which itself consists of these cubes.

The discussions in this chapter will not be based on specific sequences. Instead, they will generally deal with  $Q_w$ , the set of all possible sequences of length  $l$  and weight  $w$ , and more specifically with  $\bar{q}_w$ , the average sequence of the set  $Q_w$ .

By definition, the average sequence  $\bar{q}_w$  of length  $l$  is generated by a set of cubes composed of  $\bar{T}(k)$   $k$ -cubes for each value of  $k$  ( $k = 0, 1, \dots, n'$ ), where  $\bar{T}(k)$  is the average number of  $k$ -cubes over all  $q_w$ 's,  $\forall q_w \in Q_w$ . For instance,  $\bar{T}(2)$  corresponds to the average number of 2-cubes over all the possible sequences in set  $Q_w$ .

Let us also define  $\bar{T}$  as the total number of cubes of sizes  $k=0, 1, \dots, n'$ , that are needed to generate  $\bar{q}_w$ , thus:

$$\bar{T} = \sum_{k=0}^{n'} \bar{T}(k) \quad (5.1)$$

Having defined  $\bar{T}(k)$  as the number of cubes of size  $k$  needed to generate  $\bar{q}_w$ , we will proceed to demonstrate how to find it in general.

Given the sequences of length  $l = 2^{n'}$ , the total number of sequences with weight  $w$  is obtained from the number of possible choices of  $w$  out of  $l$ :

$$|Q_w| = \binom{l}{w} \quad (5.2)$$

It is known that in an  $n'$ -variable function, each one of the  $2^{n'}$  vertices has  $n'$  adjacents. Consequently, the number of distinct  $k$ -cubes which include a fixed vertex is  $\binom{n'}{k}$ , since  $k$  out of the  $n'$  adjacents of this vertex will be included in each  $k$ -cube. For example, the number of all possible 2-cubes that includes a specific vertex in a 6-variable function is  $\binom{6}{2}$ . This number should be multiplied by  $2^{n'}$  (the total number of vertices) in order to determine the number of possible  $k$ -cubes in an  $n'$ -variable function. However, it should be divided by  $2^k$  as well, or else the same  $k$ -cube will be considered with each of its  $2^k$  vertices. Thus:

$$C(k) = \binom{n'}{k} 2^{n'-k} \quad (5.3)$$

Recall that given a function, a cube is said to be essential if it is considered indispensable in the sum-of-products representation of that function.  $N(k)$  is defined as the number of sequences in  $Q_w$  in which a fixed  $k$ -cube is essential.

Finally, to obtain  $\bar{T}(k)$ , the total number of possible  $k$ -cubes in  $Q_w$  is divided by the cardinality of  $Q_w$  [Mileto 64]. Thus we have,

$$\bar{T}(k) = \frac{C(k) \cdot N(k)}{|Q_w|} \quad (5.4)$$

$C(k)$  and  $|Q_w|$  (in terms of  $n'$  and  $k$ ) are substituted from (5.3) and (5.2) respectively. However,  $N(k)$  is determined in the following sections, 5.2 and 5.3, in order to be substituted also in equation (5.4). In section 5.4 the average number of  $k$ -cubes are evaluated, and in section 5.5 and 5.6, it is shown how to calculate the amount of reduction in weight, and the consequent deception volume for a given  $p$ .

## 5.2 Essential Cubes and Isolated Vertices:

*Definition:* A fixed  $k$ -cube is considered essential (i.e. required to generate a sequence  $q_w \in Q_w$ ), if at least one of its  $2^k$  vertices is isolated. This means that this vertex is not covered by any other cube.

To determine that a certain vertex is isolated, one of the following conditions has to be fulfilled:

- all of its adjacent vertices should be assigned to 0's;
- there should not exist any cube which covers the isolated vertex and any one of its adjacent 1's together.

To clarify the derivation of the main formula needed to calculate the  $N(k)$ 's, we use the example shown in Figure(5.1), where the vertex  $a/1$  and the 1-cube including it are considered as an isolated vertex and an essential cube, respectively. However, we do not limit our discussions to the size of the function and the essential cube shown in Figure(5.1). Hence, we consider the set  $Q_w$ , in general, and assume that the vertex  $a/1$  happens to be isolated vertex of it. Therefore its  $k$ -cube is considered as an essential cube. Our target is first to find out the total number of functions in  $Q_w$  which contain the fixed  $k$ -cube.

It is clear that  $a/1$  has  $k$  adjacent vertices inside the  $k$ -cube, and  $n' - k$  adjacent vertices outside of it. The ones outside are denoted as  $b$ -vertices, and it is assumed that  $j$  of them are assigned to 1's (where  $0 \leq j \leq n' - k$ ), and the rest  $n' - k - j$  to 0's. We introduce two other types of vertices:

- a  $c$ -vertex is the adjacent vertex common to a pair of  $b/1$ 's;
- $d$ -vertex is a vertex adjacent to a  $b/1$ -vertex, but which is not a  $c$ -vertex nor  $a/1$ .

The cardinality of the set of  $d$ -vertices is  $n' - k - j$ .



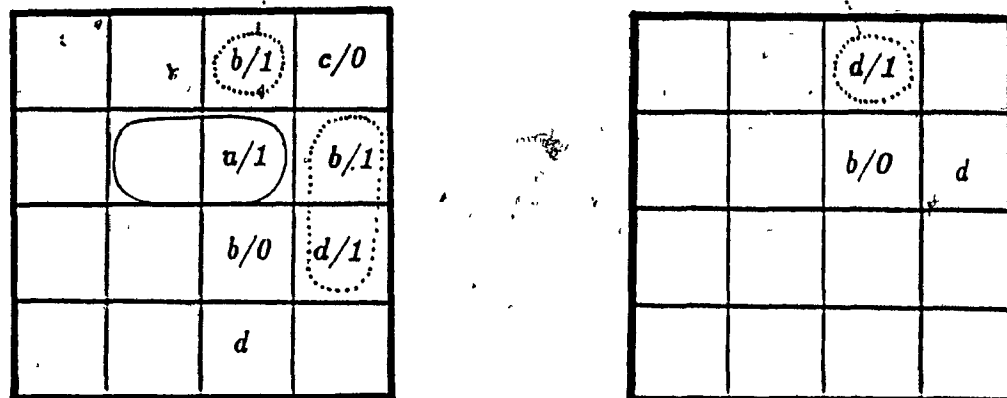


Figure (5.1) Isolated Vertices in an Essential Cube

Since vertex  $a/1$  is regarded as an isolated vertex, then it is required that:

- \*every  $c$ -vertex be assigned to 0 to avoid forming a 2-cube with  $a/1$  and the pair of  $b/1$ 's;
- \*at least one  $d$ -vertex, for each  $b/1$ , be assigned to 1, to provide a cube which covers  $b/1$  and  $d/1$ .

### 5.3 The Number of Sequences with Isolated Vertices in a $k$ -Cube:

Our aim here is to compute the number of strings in  $Q_w$  which have isolated vertices that are based on the conditions presented in the previous section.

If we contribute only one fixed  $d$ -vertex for each fixed  $b/1$ , the number of sequences or functions in  $Q_w$  where  $a/1$  is forced to be an isolated vertex in that  $k$ -cube can be obtained from:

$$\binom{l - v_k - 2j - (n' - k - j) - \frac{1}{2}j(j-1)}{w - v_k - 2j} = \binom{l - v_k - (n' - k) - \frac{1}{2}j - \frac{1}{2}j^2}{w - v_k - 2j} \quad (5.5)$$

where  $2j$ ,  $n' - k - j$  and  $\frac{1}{2}j(j-1)$  are the values of  $b/1$  and  $d/1$ ,  $b/0$  vertices and  $c/0$  vertices, respectively. Finally,  $v_k = 2^k$  is the number of vertices in a  $k$ -cube.

Since one  $d$ -vertex out of  $n' - k - j$  will be assigned to 1 for each  $b/1$ , the number of sequences in (5.5) increases to become:

$$\left( \frac{l - v_k - (n' - k) - \frac{1}{2}j - \frac{1}{2}j^2}{w - v_k - 2j} \right) (n' - k - j)^j \quad (5.6)$$

Let us consider the functions where the number of  $d$ 's that are assigned to 1 exceeds  $j$ , and introduce  $R$  as the number of extra  $d/1$ 's, where  $R = 0, 1, \dots, (n' - k - j - 1)j$ . Note that for every  $R$ , there is a set of distributions denoted as  $\{R\}$ . Wherein each member of this set,  $r \in \{R\}$ , represents the distribution of  $R + j$   $d/1$  vertices over  $j$  fixed  $b/1$ 's. Therefore, every  $r$  consists of  $j$  numbers  $r_1, r_2, \dots, r_j$ , which sum to  $R + j$ :

$$\sum_{i=1}^j r_i = R + j$$

Here the principle of inclusion and exclusion is utilized so that the functions with  $R > 0$  are also taken into consideration. Hence, the number of  $q_w$ 's with  $a/1$  as an isolated vertex in the  $k$ -cube, and with  $j$  fixed  $b/1$ 's, is:

$$\sum_{R=0}^{(n'-k-j-1)j} (-1)^R \left( \frac{l - v_k - (n' - k) - \frac{1}{2}j - \frac{1}{2}j^2 - R}{w - v_k - 2j - R} \right)^j \sum_{\substack{\text{all} \\ r \in \{R\}}} \prod_{i=1}^j \binom{n' - k - j}{r_i} \quad (5.7)$$

Expanding this formula to first include non-fixed  $b/1$ 's, i.e. having the possibility of choosing  $j$   $b/1$ 's out of  $n' - k$   $b$ -vertices, and then summing over all possible  $j$ 's ( $j = 0, 1, \dots, n' - k$ ), we get:

$$\sum_{j=0}^{n-k} \binom{n'-k}{j} \sum_{R=0}^{(n'-k-j-1)} (-1)^R \binom{n'-k-j-1}{w-v_k-2j-R} \sum_{\substack{\text{all} \\ r \in \{R\}}} \prod_{i=1}^j \binom{n'-k-j}{r_i} \quad (5.8)$$

Equation (5.8) is an expression for  $N(k, 1)$ , the number of  $Q_w$ 's with at least one fixed vertex isolated in the  $k$ -cube. Thus far, our calculations were concerned only with the cases where the  $k$ -cube is essential due to one vertex ( $a/1$ ). They did not take into consideration the possibility of having other isolated vertices in the same  $k$ -cube. To calculate  $N(k)$ , which is the number of  $Q_w$ 's where the fixed  $k$ -cube is essential, i.e. the  $k$ -cube has at least any one isolated vertex, the principal of inclusion and exclusion needs to be adopted, in order to consider the cases with multiple isolated vertices:

$$N(k) = \binom{v_k}{1} N(k, 1) - \binom{v_k}{2} N(k, 2) + \dots \binom{v_k}{v_k} N(k, v_k) \quad (5.9)$$

In the above equation (5.9),  $N(k, v)$  stands for the number of sequences in  $Q_w$  which have at least  $v$  fixed isolated vertices in the  $k$ -cube. The computation of  $N(k, v)$  is done by generalizing equation (5.8):

$$N(k, v) = \sum_{j_1=0}^{n'-k} \sum_{j_2=0}^{n'-k} \dots \sum_{j_v=0}^{n'-k} \prod_{p=1}^v \binom{n'-k}{j_p} \sum_{R=0}^{\sum_{p=1}^v (n'-k-j_p-1)} (-1)^R \binom{n'-k-j_1-\dots-j_v-1}{w-v_k-2\sum_{p=1}^v j_p-R} \sum_{\substack{\text{all} \\ r \in \{R\}}} \prod_{p=1}^v \prod_{i=1}^{j_p} \binom{n'-k-j_p}{r_{i,p}} \quad (5.10)$$

where every  $r \in \{R\}$  is composed of a different set of  $r_{i,p}$ 's that are restrained by the following condition:  $\sum_{p=1}^v \sum_{i=1}^{j_p} r_{i,p} = R + \sum_{p=1}^v j_p$ . The calculation of  $N(k, v)$

shown in (5.10) is the final formula used to evaluate the average number of  $k$ -cubes,  $\bar{T}(k)$ .

#### 5.4 Average Number of $k$ -Cubes with Exact Number of Isolated Vertices

Some of the vertices in a  $k$ -cube, contrarily to the isolated vertices, are covered by more than a single cube. Hence, these are called shared vertices. If a  $k$ -cube contains  $v$  isolated vertices, then the  $v_k - v$  remaining ones are shared vertices.

The calculation of  $N(k, v)$ , obtained from equation (5.10), does not consider a  $k$ -cube with a certain number of isolated vertices. Instead, it takes into account a  $k$ -cube with at least  $v$  fixed and isolated vertices (i.e.  $v, v + 1, \dots, v_k$ ). This is why we need to introduce  $P(k, v)$ , the number of  $q_w$ 's with exactly  $v$  isolated vertices in the  $k$ -cube. Figure(5.2) illustrates the difference between  $N(k, v)$  and  $P(k, v)$ . In this figure, the three main circles represent the three sets of  $q_w$ 's with a fixed cube considered essential due to the three isolated vertices  $a_1, a_2$  and  $a_3$ . The covered area in (a) represents the set of sequences which have at least two fixed vertices ( $a_2$  and  $a_3$ ) isolated. The cardinality of this set is  $N(k, 2)$ . Still in Figure(5.2),  $P(k, 2)$  stands for the sum of three cases where exactly two vertices are isolated.

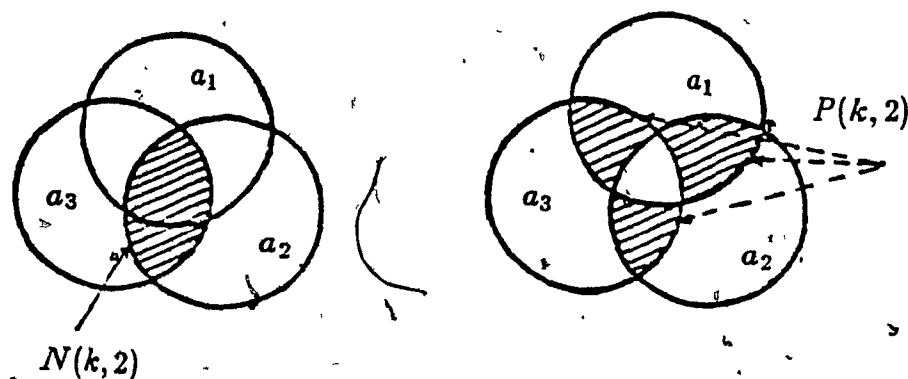


Figure (5.2) The States of Three Isolated Vertices in an Essential Cube

Thus, from the above,  $N(k)$  expressed in terms of  $P(k, v)$  is:

$$N(k) = P(k, 1) + P(k, 2) + \dots + P(k, v_k) \quad (5.11)$$

In order to calculate the value of  $P(k, v)$ , we substitute the previously derived  $N(k, v)$  and use the principal of inclusion and exclusion:

$$P(k, v) = \binom{v_k}{v} \left\{ \binom{v_k - v}{0} N(k, v) - \binom{v_k - v}{1} N(k, v+1) + \dots + \binom{v_k - v}{v_k - v} N(k, v_k) \right\} \quad (5.12)$$

Returning to  $\bar{T}(k)$ , i.e. the average number of  $k$ -cubes in  $\bar{Q}_w$ , and substituting equation (5.11) into equation (5.4):

$$\bar{T}(k) = \frac{C(k)}{|Q_u|} \{P(k, 1) + P(k, 2) + \dots + P(k, v_k)\}$$

from which we obtain:

$$\bar{T}(k, v) = \frac{C(k)}{|Q_w|} P(k, v) \quad (5.13)$$

and

$$\bar{T}(k) = \bar{T}(k, 1) + \bar{T}(k, 2) + \dots + \bar{T}(k, v_k) \quad (5.14)$$

where  $\bar{T}(k, v)$  is the average number of  $k$ -cubes in  $\bar{Q}_w$  which have exactly  $v$  isolated vertices, and thus  $v_k - v$  shared vertices. Hence, there is an amount  $v \cdot \bar{T}(k, v)$  of isolated vertices, and an amount  $(v_k - v) \cdot \bar{T}(k, v)$  of shared vertices in the entire  $k$ -cubes with  $v$  isolated vertices in  $\bar{Q}_w$ . Both of these amounts are used in the next section to find the weight of a  $k$ -cube which has exactly  $v$  isolated vertices.

### 5.5. Weight of a $k$ -Cube with $v$ Isolated Vertices:

A new value called the sharing factor is introduced in this section to determine the weight of each  $k$ -cube with  $v$  isolated vertices. We first calculate the total number,  $IV$ ,

of isolated vertices in  $\bar{q}_w$ . This number is the weight of the isolated vertices, termed as  $weight(k, v)$ :

$$IV = \sum_{k=0}^{n'} \sum_{v=1}^{v_k} v \cdot \bar{T}(k, v) \quad (5.15)$$

From this we calculate the weight of the shared vertices  $w - IV$ , and  $SV$ , the number of the shared vertices:

$$SV = \sum_{k=0}^{n'} \sum_{v=1}^{v_k} (v_k - v) \cdot \bar{T}(k, v) \quad (5.16)$$

The sharing factor  $f$  indicates the average number of cubes that cover a shared vertex:

$$f = \frac{SV}{w - IV} \quad (5.17)$$

This sharing factor assists in deriving the weight of each  $k$ -cube with  $v$  isolated vertices:

$$weight(k, v) = (v + \frac{1}{f}(v_k - v)) \cdot \bar{T}(k, v) \quad (5.18)$$

Based on the above calculation of  $weight(k, v)$  in equation (5.18), the different cubes which generate  $\bar{q}_w$  can be categorized. Hence, as explained in the following subsection, the selection of cubes depends upon the value of  $weight(k, v)$ .

## 5.6 The Available Cubes and the Reduction in Deception Volume:

A selection of a certain number ( $p$ ) of cubes out of the  $\bar{T}$  cubes which generate  $\bar{q}_w$  is needed, if the number of cubes to generate a sequence as close as possible to  $\bar{q}_w$  is limited by  $p$ , and  $p < \bar{T}$ . Such a sequence can be obtained if the first  $p$  cubes with the highest amount of  $weight(k, v)$ 's are selected. The summation of these weights results in the weight of  $\bar{s}_w$ , which is actually the amount of reduction in  $\bar{q}_w$ 's weight:

$$\text{weight of } \bar{s}_w = \sum_{\text{subset}} \text{weight}(k, v)$$

The new weight, or the weight of the modified sequence  $\bar{q}_w$  eventually is:

$$\text{weight of } \bar{q}_w = u - \text{weight of } \bar{s}_w$$

After obtaining this new weight, the reduced deception volume for an average sequence of weight  $w$  is calculated. Moreover, if various weights are considered, it is possible to generate a curve indicating the new deception volumes of average sequences as a function of  $w$ , Figure(5.3).

If the number of available cubes  $p$  is increased, the selected subset of cubes is enlarged, and therefore, the reductions in weight and deception volume are increased as well. This reveals that the deception volume can be controlled by  $p$ . Hence, we have a trade off between the size of Modifier block and the reduction in error masking desired. If our objective is to reach a specific deception volume, then the required number of  $p$  (i.e. the size of the Modifier block), for average sequences of specific lengths, can be obtained from the deception volume curves (as in Figure(5.3)).

Various lengths of average sequence with different amount of  $p$ 's have been studied. Their deception volume curves attest a significant amount of reduction in error masking. Figure(5.3) demonstrates two case studies where the strings of length  $l = 2^{12}$  and  $l = 2^{14}$  with various values of  $p$ 's are examined. For instance if three different sizes of Modifier blocks are considered, e.g.  $p = 150, 300$  and  $450$ , an average sequence of length  $l = 2^{12}$  and original weight  $w = 1500$  is reduced to  $w' = 1059, 782$  and  $524$  respectively. Thus, the reduction in deception volume is more than  $2^{500}, 2^{1000}$  and  $2^{1600}$  times, respectively.

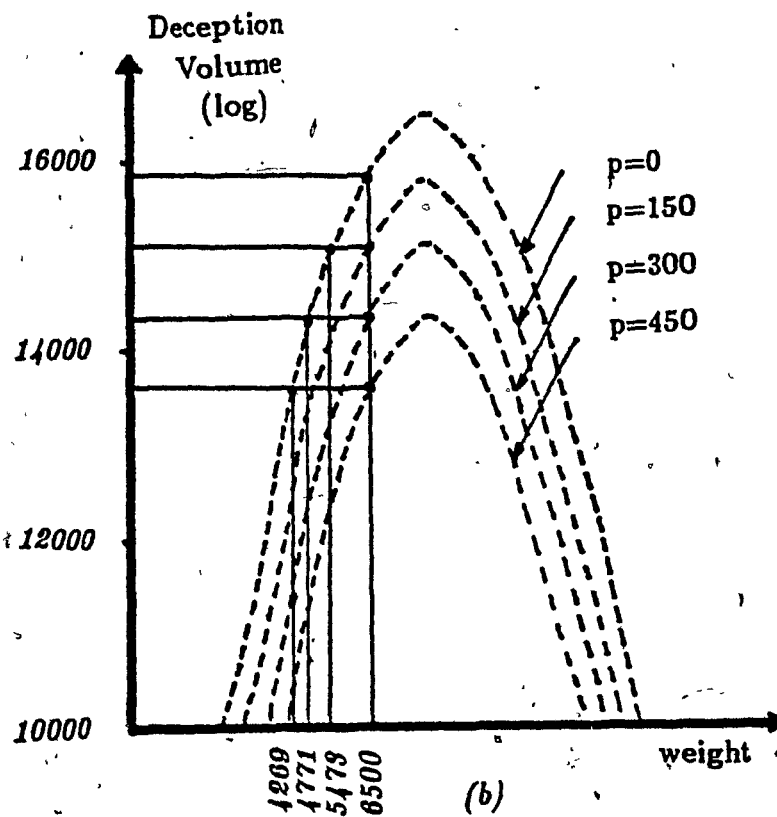
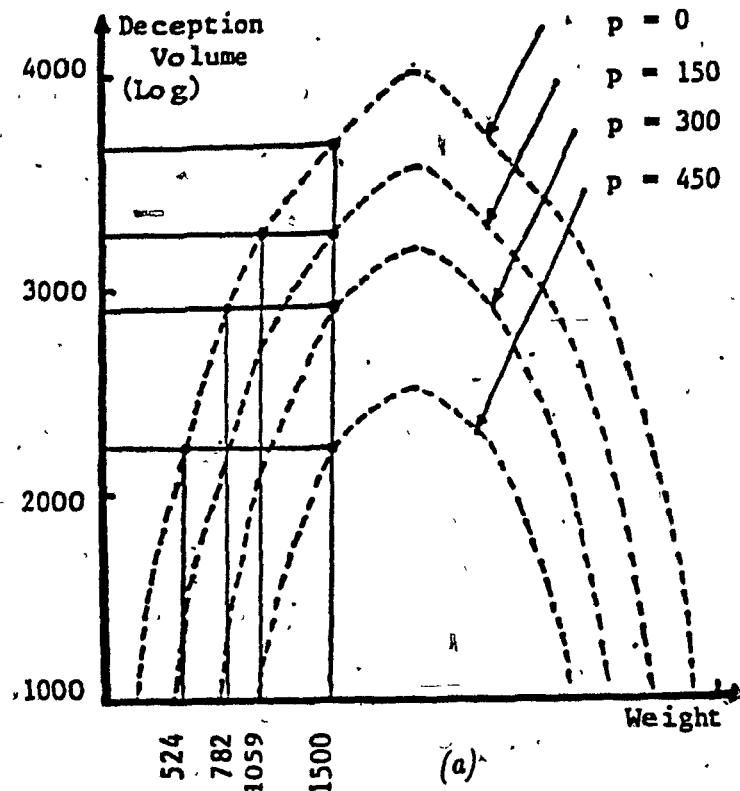


Figure (5.3) The Trade-Off Curves



The simple trade-off curve allow one to decide the optimal solution between the reduction desired versus silicon area overhead that one can achieve. Thus, IC designers are provided with a trade-off for adding more circuitry to achieve a higher assurance that error masking will not occur. No other published scheme in the literature provides such amounts of reduction with such small area overheads.

## Chapter 6

### Realization of the ODM Scheme

The previous chapters describe the motivation behind the ODM scheme, the modification concept adopted by this scheme, various methods to generate the modifier sequence, and an analytical proof which shows the effectiveness of the ODM scheme for average cases. This chapter first illustrates in section 6.1, the realization of the scheme as a general BIST model. Then in section 6.2, the circuit-specific implementation of this scheme is presented.

#### 6.1 The General BIST Design of the ODM Scheme:

The design of a general BIST model, under the ODM scheme, is illustrated in Figure(3.11). This section describes the structure and the functionality of each one of its blocks: the standard BIST facilities, the Modifier block, and the Improved Compressor block.

### 6.1.1 The Standard Self-Test Facilities:

The dotted box in Figure(3.11) represents the self-test facilities in the standard form of BIST. This box contains, in addition to the combinational CUT, an IPG and a signature register realized by an MISR. The former generates the required input patterns, and the latter is used, in this scheme, as a parallel to serial compressor (P/S Compressor).

#### 6.1.1.1 Input Pattern Generator:

The actual task of this block is to generate two sets of  $l$  distinct patterns in parallel ( $l$  being the number of pseudo-random patterns needed to achieve the desired fault coverage). These two sets are the following:

- Test Set for the CUT:

As in any standard BIST scheme this set consists of pseudo-randomly generated test patterns and is applied on the  $n$ -input lines of the CUT.

- Input Set for the Modifier:

This set is composed of  $n' = \lfloor \log(l) \rfloor$  bit wide patterns (where  $n' \leq n$ ), and is utilized as the input combinations (at most  $2^{n'}$ ) to the modifier function which generates  $s$ .

Recall that the test length  $l$  needed to provide the desired coverage can be determined according to one of the techniques described in section 2.2.2.1.

Since an LFSR, whose largest appeal lies in its capability to generate pseudo-random patterns, is to date the most efficient type of test pattern generator in BIST, the following two designs, which are proposed to realize the IPG block under ODM, are based on LFSR's and generate the two sets of patterns:

A. The first design proposed is that of an IPG which consists of two independent maximum cycle length LFSR's, as shown in Figure(6.1).

- \* The first LFSR is of size  $n'$ , i.e. generates up to  $2^{n'} - 1$  distinct patterns. All its outputs are connected directly to the first  $n'$  inputs of the CUT, as well as to all the input lines of the Modifier block.

- \* The second LFSR is an  $n - n'$  bit long LFSR with its outputs applied to the remaining lines of the CUT.

Thus, in this case, the input set for the Modifier is provided by the first LFSR, whereas the test set for CUT is generated by both LFSR's. Importantly, such an implementation does not affect the pseudo-random nature of the input patterns for the CUT [Bhavsar 85].

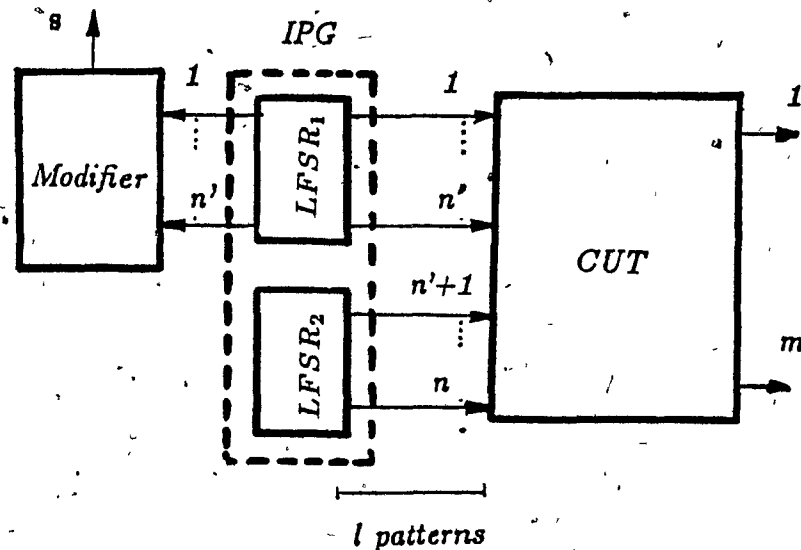


Figure (6.1) An IPG Design with Two LFSRs

B. The second proposed design, shown in Figure(6.2), is based on a pseudo-exhaustive input pattern generation technique. As discussed earlier, the important advantage

provided by these techniques is the ability of generating exhaustive patterns on  $n'$  output lines in an  $n$  stage LFSR. One such technique [Tang 84] [Barzilai 83] [Wang 84] [Wang 86b] has to be adopted to realize an IPG with this advantage. A selection among the several techniques presented in section 2.2.1, must take into consideration the three factors:  $n$ ,  $n'$  and  $l$ . The LFSR shown in Figure(2.4) is an example of such an IPG. It has  $n = 4$  output lines connected to the CUT, and one of its subsets of  $n' = 3$  lines is connected to the Modifier block.

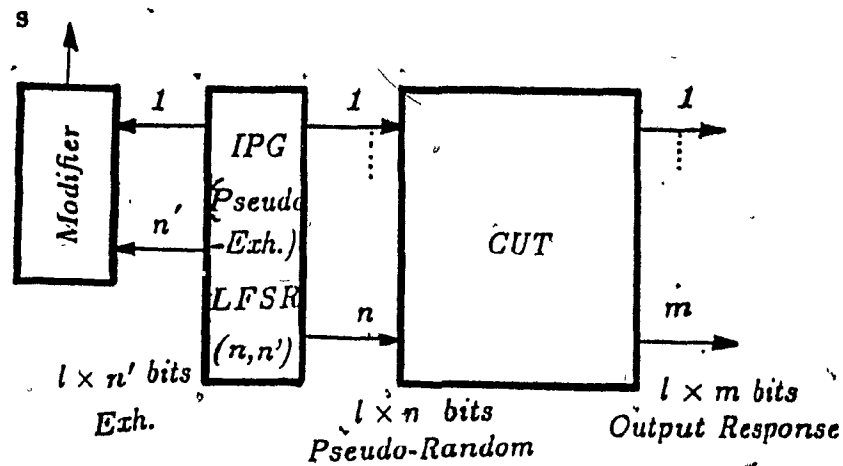


Figure (6.2) An IPG Design Using a Pseudo-Exhaustive Technique

#### 6.1.1.2 P/S Compressor:

If the modification concept is applied to every single line of a circuit with  $m$  output lines, then  $m$  Modifier blocks are required. However, for large values of  $m$ , such an approach is clearly very expensive. To solve this problem, we propose the use of a P/S Compressor which transforms every  $m$ -bit output vector into a single bit.

The selection of a proper P/S Compressor is crucial to the desired effectiveness of error coverage. There are various possibilities which can be used to select a proper P/S Compressor. In the following, four schemes with their advantages and disadvantages are described:

- The simplest P/S compression possibility is to use any one output line to create the serial string and to ignore all the others. However, this is clearly inappropriate because many faults localized in certain areas of a faulty IC chip may give completely correct results on the selected output line.
- Another P/S compression possibility is to sample the output lines to create a single stream of bits. This is also inappropriate because certain parts of an IC chip operate only when control variables switch them on and switch other parts off, therefore, it is possible to have bursts of incorrect values on certain output lines when these sections are switched on. Such burst of erroneous bits may occur between the sampling times that of the output lines, and therefore, may possibly be absent from the serial stream.
- Yet another possible P/S Compressor is a parity tree (mentioned in section 2.3.2). This also is generally inappropriate because the output of a parity tree compresses each m-bit pattern independently of the preceding patterns. Therefore, the faults of an IC which can only be detected by one or few patterns have more chances of not being detected, compared to the situation where the result of the P/S compression depends on the latest applied pattern as well as on the preceding ones.
- The final type of P/S compressor that we consider is the MISR. At each clock cycle, an MISR produces a quotient bit as well as an m-bit remainder that together represent the last pattern applied to the MISR as well as the preceding ones. Therefore,

this property of the MISR makes it a better choice for P/S compression than the three above possibilities.

Based on the above discussions of the different P/S Compressors, an MISR is used as a P/S Compressor due to its overall superiority. The compression that we use the MISR for, in the ODM scheme, does not specifically consider the remainder or final contents of the MISR as it is done in signature analysis. Instead, we monitor the quotient bit of the MISR and use this sequence as the compressed sequence. This compressed sequence is the one that is then fed to the Improved Compression stage. Hence, the loss of information due to the MISR in this application is not the same as in the case of signature analysis. The loss of information, here, arises from mapping the  $l \times m$  bit matrix to an  $l$ -bit serial string  $q$ . Since this mapping involves only GF(2) additions, it is uniform. Therefore, there is an equal number of  $l \times m$  matrices out of the number of all possible matrices,  $T$ , that map to the same serial string. This number is  $T/2^l$ . If we assume that each of the  $T$  possible matrices is equally likely, the error masking probability introduced by the MISR (P/S Compressor) is  $pr_{p/s} = 2^{-l}$ . Since  $l$  is usually of the order of thousands or millions, the loss of error coverage at this stage is not very significant.

Two other advantages of adopting an MISR as a P/S Compressor are the following. First, the final content of the MISR can be utilized for further error coverage enhancements if needed. This is not true for a parity tree P/S Compressor which has no contents at the end of compression. The other advantage of using an MISR is that it allows the implementation of the ODM BIST model on any existing MISR-based BIST scheme by simply adding the Modifier and the Improved Compressor blocks.

### 6.1.2 The Modifier Block:

The modifier string generation method used for the Modifier block is the one which generates sequences by totality. The Modifier block, in this case, consists of  $p$  product terms predetermined in advance to generate the modifier strings. These product terms, as well as the ones in the original quotient function  $F_q$ , are represented throughout this chapter by cubical notations [Dietmeyer 79]. Since the Modifier block realizes a multiple input and single output fixed logic function, therefore a very convenient implementation for it (see section 4.3) is the logic function block model used in [Mead 80]. The latter model consists of a set of transistors that fully decode the input combinations of  $n'$  input variables. However, in our application, not every possible product term is built, but only the ones predetermined in advance. It is because of the hardware constraint that this function block is limited to  $p$  product terms only. If an input pattern is covered by one or more product terms of the logic function, then the output for this pattern is a logic one, or else a logic zero. In order to determine the functionality of this block, i.e. the set of product terms, a specific procedure is described in section 6.2.1. After applying this procedure, the resulting product terms are used to set (i.e. hardware program) the Modifier block. Example(6.1) illustrates the realization of such a block.

The size of the Modifier block depends simply upon  $p$  and  $n'$ . Thus, it is application specific. As shown in section 5.6, this flexibility is one of the advantages of the ODM scheme, because it provides the possibility of increasing the effectiveness of the scheme by increasing the number of product terms available. Furthermore, the layout estimates of the function logic block based on [Mead 80] show that a Modifier block which can implement  $p = 150$  cubes with  $n' = 16$  input variables requires a silicon area which corresponds to the size of a 16-bit MISR, when the latter is based on the shift register



designs in [Newkirk 83]. Thus, a large number of  $s$  sequences can be generated with a *Modifier* block of relatively small area.

Discussions about the circuit-specific implementation of the ODM scheme in general, and detailed descriptions about determining the functionality of this block are presented in section 6.2.

### 6.1.3 The Improved Compressor:

The improved compression block performs the two most important operations in the ODM scheme; the output data modification and the non-uniform compression. Thus, its hardware design includes an *Exclusive-OR* gate to realize the former operation, and a count-based compressor (i.e. a *Counter for One's Counting*) to perform the latter one.

The two data streams, the *modifier string*  $s$  and the *quotient-bit string*  $q$ , enter this block to be *Exclusively-ORed* to produce the  $l$ -bit long modified string  $q'$ . The new string  $q'$  enters the *Counter* to be compressed into an  $n'$ -bit long signature. This is the compression step that has to be performed by a non-uniform technique, otherwise the modification will not serve any purpose. The use of the best possible scheme can be judged by the reduction in the deception volume. For instance, if the compression is done by *One's Counting*, and if the number of ones in  $q$  is  $w$  and in  $q'$  is  $w'$ , then the improvement in deception volume is given by the ratio:

$$\frac{pr'}{pr} = \frac{\binom{l}{w'}}{\binom{l}{w}}$$

where  $pr'$  ( $pr$ ) is the error masking probability obtained by compressing  $q'$  (respectively  $q$ ). Similarly, if the compression is done by *Transition Counting*, and there are  $t$  and  $t'$  transitions in  $q$  and  $q'$  respectively, then the improvement ratio is

$$\frac{pr'}{pr} = \frac{\binom{l-1}{t'}}{\binom{l-1}{t}}$$

The respective formula for the Edge Counting case is:

$$\frac{pr'}{pr} = \frac{\binom{l+1}{2e'-1}}{\binom{l+1}{2e-1}}$$

where  $e$  and  $e'$  are the number of edges in  $q$  and  $q'$ , respectively. Similar formulae can be obtained for other count-based techniques.

In general, if it is possible to replace a counter by an LFSR to perform the same task, a gain in terms of hardware overhead is obtained since an LFSR design requires typically less area than a counter. Such replacement is realizable in the ODM model because the signature in our Counter which performs the count-based compression does not necessarily have to be the expected number of counts itself. In fact, it suffices to have any signature provided that it indicates that the Counter has performed the expected number of counts. Thus, we will adopt the use of an LFSR as a Counter in the Improved Compression block. In our design, this is realized by adopting an autonomous LFSR (section 2.1.1) with the string under count  $q'$  applied as a control signal enabling the LFSR clocks to change its state for each count. For instance, if the modified string to be compressed is  $q' = 00010010$ , and if the LFSR is initialized to 110, as in Figure (6.3), then although the final state would be 111, the LFSR will indicate that the Counter has performed  $w^* = 2$  counts. It has to be indicated that an LFSR under this use must be a maximum cycle length LFSR with  $n'$ -stages (i.e. cycles through  $2^{n'} - 1$  distinct states) to guarantee that no repetition of states will occur during the application of any  $l$  bit long  $q'$ , since  $l \leq 2^{n'}$ .

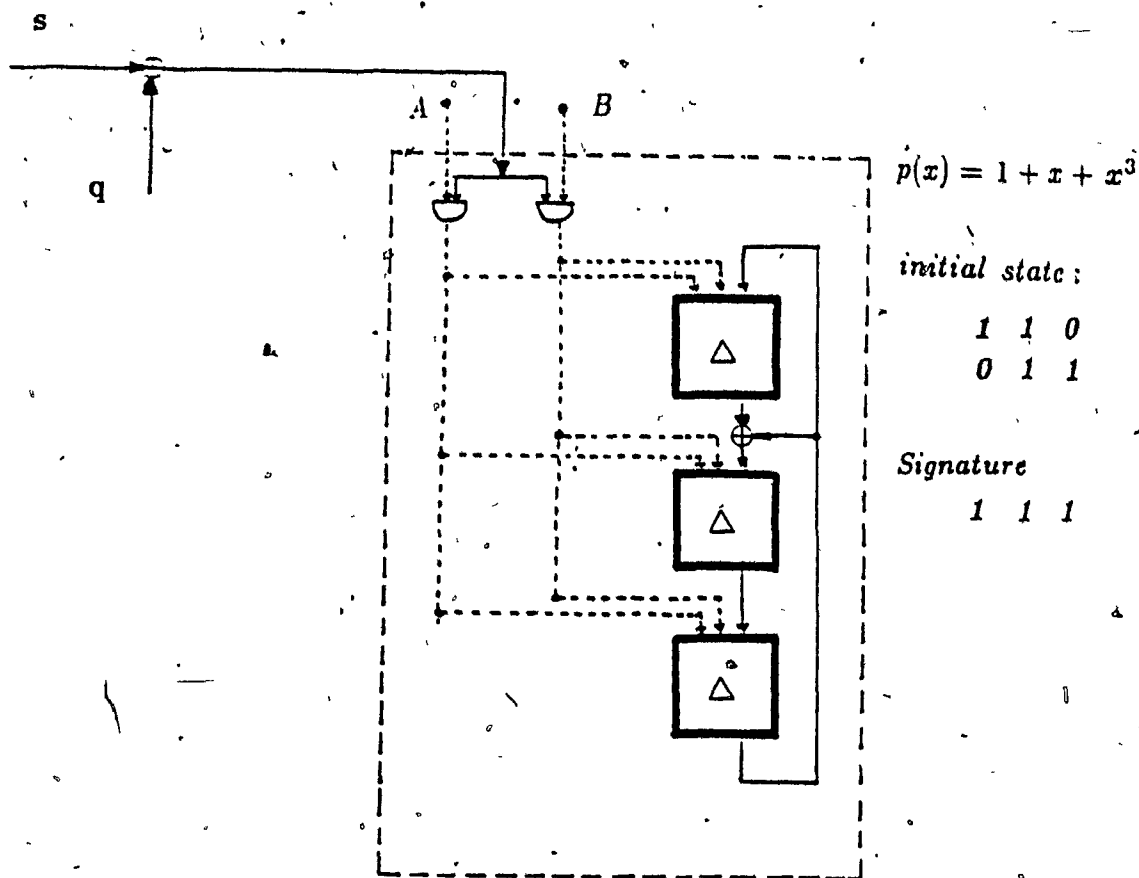


Figure (6.3) The Improved Compressor

Moreover, notice that given the expected number of counts and the characteristic polynomial, it is possible to force such an LFSR to any predetermined final state [McAnney 86]. If we let the predetermined final state be a regular sequence, e.g. a string of all 1's or a string of alternative 1's and 0's, then the important benefit obtained is the possibility of eliminating the whole Reference block used to store the fault-free signature. This elimination is possible because a Comparator can perform the verification of such a regular sequence with no Reference block dedicated for this purpose.

Since the output data compression, under the ODM scheme, is performed by two succeeding compressors, the P/S Compressor and the Improved Compressor, the error masking probability of the entire compression includes both compressors. Hence:

$$pr_{tot} = 1 - (1 - pr_{p/s})(1 - pr_{count})$$

Thus, an optimal  $pr_{tot}$  is the result of minimal  $pr_{p/s}$  and minimal  $pr_{count}$ . But since  $pr_{p/s} = 2^{-l}$  (derived in section 6.1.1.2) is already an insignificantly small amount, it suffices to provide an important reduction in  $pr_{count}$  to have an optimal  $pr_{tot}$ . In fact, the tremendous improvements in error coverage provided by the ODM scheme are the result of obtaining major reductions in  $pr_{count}$ .

## 6.2 Circuit-Specific Implementation

Generally, the implementation of a standard BIST scheme requires some knowledge about the CUT's functionality. This is usually needed to:

- determine the number of necessary pseudo-random patterns  $l$ . The determination of  $l$  can be done by one of the methods mentioned in section 2.2.2.1. If necessary a technique to detect the random pattern resistant faults may be utilized (see section 2.2.2.2).
- identify the characteristics of both polynomial dividers: the LFSR as IPG, and the MISR as signature register (i.e. number of stages, characteristic polynomials, initial states).
- perform a functional simulation to determine the fault-free remainder in the MISR.

In order to implement the ODM scheme for a given CUT, one needs to perform all the above activities, except that instead of determining the fault-free remainder in the MISR, the fault-free quotient bit sequence  $q$  of the MISR is needed. In addition, some extra work described in the following section is needed.

### 6.2.1 A Procedure to Implement the ODM Scheme:

The extra work required to carry out the implementation of the ODM scheme is presented as a procedure which consists of the following steps:

1. Apply Boolean Function Minimization coupled with Cube Selection:

- Given  $q$ , the MISR's quotient bit sequence, and the corresponding set of input patterns ( $n'$ -bit long  $l$  patterns) for the Modifier block, find out the function which generates the quotient sequence  $F_q$  and its minimal sum-of-products form expression. Notice that since  $l$  is often not much larger than, say, 1 million,  $n'$  is less than 20. An exact minimization by a very efficient algorithm [Dagenais 85] is possible for such functions.
- Given the set of product terms in the expression for  $F_q$  and the size  $p$  of the Modifier block, determine the functionality of the Modifier by selecting  $p$  cubes out of  $t$ , if  $t > p$ . This selection is performed by utilizing a cube select algorithm based on covering the largest number of 1-vertices in the quotient function  $F_q$ .

2. Set the Modifier Block:

- Given  $p$  selected cubes, perform the hardware programming of this functional block [Mead 80], and determine the resulting modifier string  $s$ .

3. Determine the specifications of the Counter:

- Form an LFSR-Counter of size  $n'$  with a primitive polynomial.
- Based on  $q$  and  $s$ , compute  $w'$ , the weight of the modified string  $q'$ .
- Based on  $w'$  and the characteristic polynomial of the LFSR-Counter, find out the

initial state, given a final state (i.e. the signature), where, for example, a typical final state can be a string of alternative 1's and 0's.

The following example illustrates the above procedural steps.

**Example 6.1:** In order to maintain the simplicity of this example, the predetermined test length is limited to  $l = 64$ . Thus, the number of variables required to generate the  $l$ -bit long modifier string  $s$  is  $n' = 6$ . We assume that the serial string  $q$  is obtained from the functional simulation of a CUT and its P/S compressor (MISR). Some bits of the  $q$  that we assume, among with their corresponding  $n'$ -bit input patterns (to the Modifier), are shown in Table(6.1). The entire truth table, which results by completing Table(6.1), represents the function  $F_q$  to the quotient sequence.

Table 6.1

Modifier Inputs and Quotient Bit String

IPG's input set for Modifier	Quotient bit string
$f e d c b a$	$F_q$
0 0 0 1 0 1	1
0 0 1 0 1 0	1
0 1 0 1 0 0	1
1 0 1 0 0 0	1
0 1 0 0 0 1	1
1 0 1 1 1 0	0
0 1 1 1 0 1	1
1 1 0 1 1 0	0
. . . . .	.
. . . . .	.
. . . . .	.

Suppose that the following sum-of-product expression is obtained as a result of applying Boolean minimization to the original function  $F_q$ .

$$F_q = \overbrace{\bar{a}\bar{b}}^{4\text{-cube}} + \overbrace{\bar{b}\bar{d}\bar{f} + \bar{c}\bar{d}\bar{f}}^{3\text{-cubes}} + \overbrace{\bar{b}\bar{c}\bar{d}\bar{e} + \bar{a}\bar{d}\bar{e}\bar{f}}^{2\text{-cubes}} \\ + \overbrace{abc\bar{d}\bar{e} + abcd\bar{f} + a\bar{c}d\bar{e}\bar{f} + \bar{a}b\bar{c}e\bar{f}}^{1\text{-cubes}} + \overbrace{\bar{a}bcd\bar{e}\bar{f}}^{0\text{-cubes}}$$

The set of cubes derived from the above expression consists of 10 cubes. These are presented in Table(6.2) after having been sorted on the basis of cube size.

Table 6.2

The List of Cubes

	<u>a</u>	<u>b</u>	<u>c</u>	<u>d</u>	<u>e</u>	<u>f</u>
4 - cubes	{	0	0	x	x	x
3 - cubes	{	x	0	x	0	x
		x	x	0	0	x
2 - cubes	{	x	1	0	1	0
		1	x	x	0	1
1 - cubes	{	1	1	1	0	0
		0	1	1	1	0
		1	x	0	1	1
		0	1	0	1	1
0 - cubes	{	0	1	1	0	1

Recall that  $p$  is decided according to either the area dedicated to the Modifier block, or the required amount of deception volume. In this example, we consider different sizes of a Modifier block by assigning  $p$  to the values, 3, 6, 8, and 10. The selection step here

is performed on the basis of cube size, i.e. the larger cubes are selected first. The reason is that the larger the size of cube is, the higher the number of 1-vertices it covers in the original function  $F_q$ . An efficient cube select algorithm is developed in the following section, and applied to this same example. As a result of such a cube selection, the modifier function, i.e. the set of selected cubes, for each value of  $p$  is determined. A convenient way to visualize the hardware realization of these functions is presented in Figure(6.4), where each cube is identifiable by its hardware programmed points.

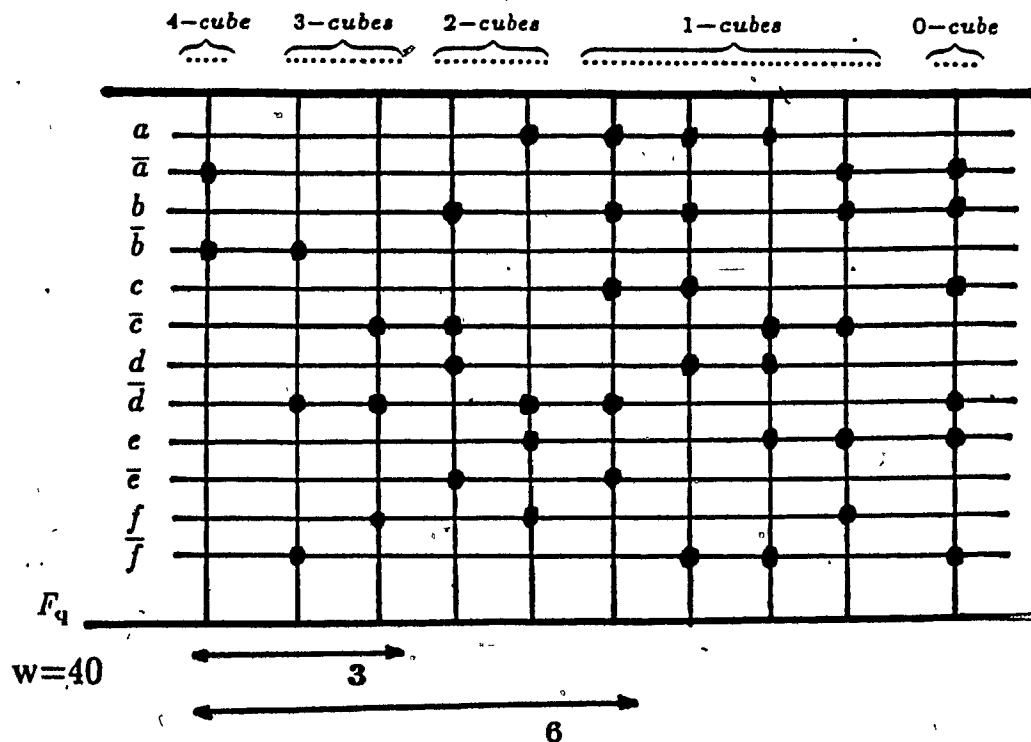


Figure (6.4) The Hardware Realization of the Modifier Block

Table(6.3) illustrates the changes in weight and deception volume for the different values assigned to  $p$ . For instance, if the number of cubes is limited to 6, the weight of the modified string  $q'$  is reduced from 40 to 6; whereas the deception volume decreases from  $1.4 \times 10^{19}$  to  $5.2 \times 10^8$ .



**Table 6.3**

**Reductions in Weight and Deception Volume**

# of cubes	weight	Deception Volume
0	40	$1.4 \times 10^{19}$
3	14	$1.8 \times 10^{15}$
6	6	$5.2 \times 10^8$
8	2	$2 \times 10^3$

Furthermore, this simplified example, apart from its basic illustrative objective, confirms the flexibility and the effectiveness of the ODM scheme.

**6.2.2 Selection of a Limited Number of Cubes:**

It was discussed previously that, in general, due to the hardware constraint in BIST, the area overhead needed to realize a Modifier block is bounded by limiting the number of cubes that form the modifier function. Therefore, the modifier function which generates the sequence  $s$  consists of at most of  $p$  cubes. However, following the ODM concept,  $s$  is intended to be as close as possible to the original quotient sequence  $q$ . Therefore, the problem is to select  $p$  out of a given set of  $t$  cubes, which cover a high number of 1-vertices in the expected function  $F_q$ . To do so, a selection process is needed.

The simplest selection process, as shown in the previous example, is the one based merely on the sizes of cubes. This consists of selecting the largest cubes first, since the larger the size of a cube is, the higher the number of 1-vertices it covers. For instance, the 3-cube  $xx00x1$ , in a six variable function, covers eight vertices; whereas the 2-cube  $1xx011$  covers only four vertices. However, to obtain modifier functions that are closer

to an  $F_q$ , processes more efficient than the selection of the largest  $p$  cubes, might be needed. A heuristic algorithm for this purpose is developed and presented in this section along with an application example.

This heuristic algorithm is called *cube select algorithm*. Prior to selecting a cube, it utilizes useful evaluation techniques to determine the amount of contribution of a given cube in forming the modifier function. Actually, the contribution of each cube, ( $cube_i$ ), if it is added to the *selected\_set\_of\_cubes*, consists of the augmentation in the number of 1-vertices that  $cube_i$  provides to the modifier function. Such an augmentation of 1-vertices, in our scheme, is called *gain*, since it makes the modifier sequence  $s$  obtained after adding  $cube_i$  become closer to  $q$ , and therefore makes the weight of  $q'$  closer to zero. This gain is measured by the number of 1-vertices added to the modifier function due to  $cube_i$ , and is termed as *gain\_of\_cube<sub>i</sub>*. The basic task of the *cube select algorithm* is evaluating *gain\_of\_cube<sub>i</sub>*'s and accordingly selecting the first  $p$  cubes.

Even though the *gain\_of\_cube<sub>i</sub>* is based on the *size\_of\_cube<sub>i</sub>*, in effect this gain depends on the set of cubes selected earlier as well. This is because  $cube_i$  might happen to share vertices with other cubes that have already been selected. In the case of such sharing, the gain provided by  $cube_i$  is reduced, i.e. is less than the *size\_of\_cube<sub>i</sub>*. The *cube select algorithm* takes this aspect into consideration. Hence, it evaluates the *gain\_of\_cube<sub>i</sub>* before for selecting  $cube_i$ , in order to provide a modifier function which is closer to  $F_q$ .

To further increase the number of possible modifier functions, another important improvement is considered in the *cube select algorithm*. This improvement consists of extending the set of cubes from which the selection is made to one larger than the original set of  $t$  cubes. The extended set includes, in addition to the original  $cube_i$ 's, the doubles of each  $cube_i$ , i.e. the union of  $cube_i$  with each of its adjacent cubes. More

specifically, with this extension the *cube select algorithm* evaluates each *cube<sub>i</sub>*, as well as all its possible doubles before considering for selection. This means that it will not only consider the simple gain of each *cube<sub>i</sub>*, but also consider the amount of gain obtained if any *double\_of\_cube<sub>i</sub>* is selected. For instance, if the *cube<sub>i</sub>* in question is *xx00x1*, then the gain obtained from each of its three adjacent cubes *xx10x1*, *xx01x1* and *xx00x0* is evaluated first, and then added to the *gain\_of\_cube<sub>i</sub>*. From these evaluations, the cube which causes the largest gain (either the simple *cube<sub>i</sub>* itself, or one of the *double\_cube<sub>i</sub>*'s) is adopted (*adopted\_cube*) to enter the selection step. Note that it is possible for the *gain\_of\_cube<sub>i</sub>* might be larger than any *gain\_of\_double\_cube<sub>i</sub>*. This happens if the gain provided by *cube<sub>i</sub>*'s corresponding *adjacent\_cube<sub>i</sub>* is negative. By negative gain we mean that the 1-vertices, that will be added to the modifier function if this *adjacent\_cube<sub>i</sub>* is selected correspond to a higher number of 0-vertices than 1-vertices on  $F_q$ . Double cubes which contain *adjacent\_cube<sub>i</sub>*'s that yield negative gains are not adopted to enter the selection step, since in such cases, the simple *cube<sub>i</sub>*'s provide higher gain.

After evaluating the *gain\_of\_cube<sub>i</sub>* and the *gain\_of\_double\_cube<sub>i</sub>*s, the *adopted\_cube* cannot automatically enter the selection step since a further condition has to be satisfied as well, as shown in the algorithm. This condition is that the amount of gain that an *adopted\_cube* provides must be larger or equal to a predetermined reference value (limit). If this gain is less than the required limit, then the *adopted\_cube* is labeled with its gain (*label\_of\_cube<sub>i</sub>*), and kept to be re-considered later. Such a reconsideration is performed if the limit is later reduced to a value less or equal to its *label\_of\_cube<sub>i</sub>*. It has to be indicated that the value of limit is determined according to *k*, the size of cubes under consideration, and also according to the number of cubes that are needed to reach the bound *p*.

**Algorithm 4: Cube Select Algorithm:**

```

begin
  get original_set_of_cubes, and p;
  initialize size k to  $n' - 1$ ;
  initialize limit to  $2^k - 2$ ;
  initialize number_of_selected_cubes to 0;
  while ( $k > 0$  and number_of_selected_cubes  $< p$ ) do
    begin
      for each cubei do
        if (size_of_cubei = k or label_of_cubei  $\geq$  limit)
          and (number_of_selected_cubes  $< p$ )
            then
              [Cube Evaluation Section]
              count gain_of_cubei : cubei # set_of_selected_cubes
              set gain to gain_of_cubei;
              set adopted_cube to cubei;
              for each adjacent_cubej do
                begin
                  count gain_of_adjacent_cubej :
                    (adjacent_cubej # set_of_selected_cubes)  $\cap$  (original_set_of_cubes);
                  set gain_of_double_cubej to gain_of_adjacent_cubej + gain_of_cubei;
                  if (gain_of_cubei  $<$  gain_of_double_cubej)
                    then
                      set gain to gain_of_double_cubej;
                      set adopted_cube to cubei  $\cup$  adjacent_cubej;
                end;
              if (gain  $\geq$  limit)
                then
                  [Cube Selection Step]
                  set selected_cubes to selected_cubes  $\cup$  adopted_cube;
                  increment number_of_selected_cubes;
                else set label_of_cubei to gain;
            if (limit =  $2^k - 2$ ) then decrement k;
          for each cubei do
            if (size_of_cubei = k and is not selected) or (label_of_cubei  $\geq 2^k - 2$ )
              then increment number_of_potential_cubes;
            if (number_of_potential_cubes  $\leq p$  - number_of_selected_cubes)
              then set limit to  $2^k - 2$ ;
            else set limit to  $\frac{\text{limit} + 2^k - 2}{2}$ ;
          end
        end
      end
    end
  end
end.

```

Most of the operations used in the cube select algorithm to perform the evaluation steps and update the set of selected\_cubes and the original\_set\_of\_cubes are well known cube operations [Dietmeyer 79]. In other respects, the different steps used to evaluate each cube<sub>i</sub>, and decide if a cube has to be selected, or labeled and put on hold, are included in the Cube Evaluation Section of the algorithm. The steps in the cube evaluation section are applied on each cube<sub>i</sub> in the original\_set\_of\_cubes that is over a certain size  $k$ , until the number\_of\_selected\_cubes reaches  $p$ . Hence, this algorithm evaluates the larger cubes first by accepting only the cubes over size  $k$ , where  $k$  is initialized to its largest possible value  $n' - 1$ . If all the cubes of a certain size  $k$  are evaluated, and the number of selected cubes remains under  $p$ , then  $k$  is decremented and the corresponding reference value (limit) is updated so that more cubes of lower sizes are evaluated and hence selected. This process is successively repeated until all the  $p$  cubes needed for the Modifier block are determined.

#### Example 6.2:

This example illustrates the use of the cube select algorithm, by applying it on the same original set of cubes as in Example(6.1). This algorithm is used to select  $p=8$  cubes out of the  $t=10$  cubes shown in Table(6.2). The largest cube in this example, as shown in Table(6.2), is 00xxxx. Therefore, this cube is the first to enter the Cube Evaluation Section. Since 00xxxx is of size  $k = 4$ , the minimum gain possible to select this cube is  $limit = 14$ .

Table 6.4

Cube Select:  $k=4$

cube <sub>i</sub>	gain_cube <sub>i</sub>	adj_cube <sub>i</sub>	gain_adj_cube <sub>i</sub>	gain_double_cube <sub>i</sub>	selected_cube <sub>i</sub>
00xxxx	16	10xxxx	0	16	00xxxx
		01xxxx	-4	12	

The evaluation of 00xxxx, shown in Table(6.4), demonstrate that the gain obtained by it (the simple cube) is 16; whereas the gain provided by its adjacent cube 10xxxx is 0, and, therefore, the corresponding gain\_of\_double\_cube, = 16; while the gain provided by the second adjacent cube 01xxxx is a negative amount of gain -4, therefore, the corresponding gain\_of\_double\_cube, is 12. Since neither of the double\_cube,s in this case provide higher gains than the simple cube, then 00xxxx is the one selected.

The two cubes of size  $k = 3$  are evaluated using the same procedures. This is shown in Table(6.5). The limit in this case is 6.

Table 6.5

Cube Select:  $k=3$

cube <sub>i</sub>	gain_cube <sub>i</sub>	adj_cube <sub>i</sub>	gain_adj_cube <sub>i</sub>	gain_double_cube <sub>i</sub>	selected_cube <sub>i</sub>
x0x0x0	4	x1x0x0	-4	0	x0x0xx
		x0x1x0	-4	2	
		x0x0x1	2	6	
xx00x1	4	xx10x1	0	4	labeled
		xx01x1	0	4	
		xx00x0	-4	0	

In the case of cube x0x0x0, in Table(6.5), one of the doubles provides higher gain than the simple cube itself ( $6 > 4$ ). Therefore, x0x0x0 is added to the set of selected cubes. However, in the case of xx00x1, it is labeled and put on hold. Since the amounts of gains it or its doubles provide are lower than the required limit.

In order to select the final cube needed to reach  $p = 3$ , the cubes of size  $k = 2$  are considered. Table(6.6) shows that the first 2-cubes in Table(6.2) is selected, since the gain it provides is equal to the minimum required limit 4.

Table 6.6

Cube Select:  $k=2$

$cube_i$	$gain\_cube_i$	$adj\_cube_i$	$gain\_adj\_cube_i$	$gain\_double\_cube_i$	$selected\_cube_i$
$x1010x$	4	$x0010x$	-2	2	$x1010x$
		$x1110x$	-2	2	
		$x1000x$	0	4	
		$x1011x$	0	4	

This simple example demonstrates the use of cube select algorithm. Moreover, it shows that due to the different techniques adopted to evaluate the first  $p$  cubes, a more efficient set of cubes is obtained compared to the previous approach which simply selects the largest  $p$  cubes. Therefore, it results an  $s$  closer to  $q$ .

This section has shown that the circuit-specific implementation of the ODM scheme can easily be realized by utilizing software tools to perform the function minimization and cube selection processes. Such tools have been developed and used for real combinational circuit simulations, the results of which are shown in section 7.1.

### 6.3° Conclusive Remarks:

Whenever a standard BIST scheme is unsatisfiable in providing the quality of test desired, the ODM scheme can be conveniently implemented by simply adding the two extra blocks: the Modifier and the Improved Compressor. The area overhead needed to realize this augmentation consists of the extra hardware used by the Modifier block only, and not the Improved Compressor. This is because, the latter, as shown in section 6.1.3, has the advantage of being capable of eliminating the Reference block. Such elimination is possible provided that its expected signature is predetermined to be a regular one.

Furthermore, since the size of the *Modifier* block depends on the desired quality of test, the extra hardware needed is considered flexible.

Since, one of the major concerns in any *BIST* scheme is to reduce the amount of extra circuitry required, instead of adding entire self-test blocks to the original *CUT*, it is possible, in our case, to make use of the memory elements already in the circuit by redesigning them such that during the self-test mode they can function as some of our self-test blocks. For example, the existing memory elements of a *CUT*, under *ODM* scheme, can act, after reconfiguration, either as *LFSRs* to form an *IPG* or a *Counter*, or as an *MISR* to form a *P/S Compressor*. However, the functional block known as the *Modifier* has to be entirely added to the original circuit, since it is composed of a circuit-specific combinational circuitry.

In other respects, the following question still needs to be answered: although no other published scheme in the literature can provide such large improvements in error coverage for such small area overheads, what happens if the self-test facilities themselves become faulty? In fact, if there is a defect in the *IPG*, the *P/S Compressor* or in the *Improved Compressor*, then this can be detected by scanning the above shift registers. Therefore, these self-test blocks are added to the scan chain. Finally, a single fault in the *Modifier* which is shown to be a combinational block, can be detected by the exhaustive set of patterns that are applied on this block to generate *s*. The signature in the *Improved Compressor* block at the end of the self-test process has almost no chance of being correct, in this case, since the *Modifier* block make an integrated part of the entire circuit during the self-test mode, and thus is represented in the signature. In the advent of faults in the self-test circuitry, these should be detected, and hence the original *CUT* declared defective and thus replaced.



## Chapter 7

### Simulation Results and Conclusions

#### 7.1 Introduction:

This dissertation covers the theory and an application of the output data modification technique, with emphasis on the design and analysis of the ODM scheme for BIST. To illustrate the importance of this subject, the first chapters demonstrated, through a review of the previous work in the built-in self-test area, that any BIST scheme must provide a high fault coverage with low area overhead to be considered effective. This can be realized by first determining an appropriate test set able to provide the desired fault coverage, and then eliminating the risk of losing this level of coverage due to the area overhead constraint. In other words, if a test set that yields the desired fault coverage is provided, the major problem is to maintain this coverage without incurring a large amount of additional hardware. To be more specific, if output data compression is

used, then the reduction in fault coverage somehow has to be recovered through the use of a small amount of hardware. A good solution for this problem has become crucial, particularly for large VLSI chips under high test quality requirements.

Numerous papers have addressed this problem and suggested various ideas that yield limited amounts of test quality improvements (section 3.3). The solution presented in this work, as it was shown in the previous chapters, provides a tremendous improvement in fault coverage that no other published scheme in the literature can provide for the same amount of extra hardware. Moreover, the analytical proof in chapter 5 demonstrates that this scheme is very effective for any average circuit since it leads to tremendous reductions ( $2^{-\text{thousands}}$ ) in error masking probability. The improvements shown by the analytical proof and also through the different examples are measured, like in all other existing schemes, in terms of error coverage (deception volume or error masking probability), and not in terms of fault coverage. This is since it is still unknown how to relate erroneous output responses to their respective faults in the CUT. However, in order to provide an estimate in terms of fault coverage, actual fault simulations were performed. The results from these are provided in the following section.

## 7.2 Simulation Results:

In this section, the tangible effect of the ODM scheme on real combinational networks is shown through fault simulation results. Numerous simulations were performed on several example networks used as benchmarks at the ISCAS 1985 test session [Brglez 85]. From our simulations, the fault coverage of each network was computed for three different cases. The first is the case whose no output data compression performed in order to determine the original fault coverage. The two other cases use output data

compression with two different techniques to show the effect of each technique on the fault coverage. The first compression technique is the polynomial division-based one, which is usually adopted by the standard BIST schemes; whereas the second compression technique is the one used by the ODM scheme, which is presented throughout this dissertation. The same test sets consisting of pseudo-random patterns, are applied in all three cases of each network.

The fault coverage results of these simulations for the three cases (no compression, standard form of BIST and ODM scheme) are provided in Table(7.1). In each case, the fault coverage values are averaged over four simulation runs. The reason behind performing several runs was to observe the effect of using different characteristic polynomials for the pseudo-random pattern generator (LFSR) and for the polynomial divider (MISR). The polynomials for both the LFSR and the MISR were primitive and obtained from [Peterson 75]. Interestingly enough, for the cases studied, the adoption of different polynomials was observed to have no considerable effect on the fault coverages. Hence, the characteristics of the combinational networks appear to be dominant over the choice of characteristic polynomials. In fact, it is still an open problem how to predict the choice of particular polynomials that do affect the fault coverage of a CUT.

—Table 7.1

Simulation Results

Tested IC	# of patterns	No Cmprs.	Strd BIST	ODM
alu	256	100%	98.73%	100%
mc432	1024	99.24%	97.7%	99.24%
mc880	4096	99.36%	98.3%	99.36%
mc1355	4096	99.49%	98.53%	99.49%

The No Compression column, in Table(7.1), demonstrates the fault coverages of the pseudo-random test sets of lengths  $l$  without output data compression. The length of each network's test set is such that all the detectable faults are covered. Hence, when the fault coverage of a network does not reach the 100% level, the reason is that a certain number of random, pattern resistant faults exist in this network.

The Strd BIST column represents the test quality under the standard BIST schemes. The reduced fault coverages exhibited, in this column, are the results of error masking caused by the MISRs adopted. Notice that the effect of error masking may look reasonable if it is measured in terms of error coverage. However, the results in this column show that it is not when measured in terms of fault coverage. From this observation lies our motivation for improvement. For instance, the error masking probability of mc880 is estimated to be  $pr = 2^{-26}$  since the number of its output lines and thus the size of the MISR is  $k = 26$ . However, the real effect of masking, measured by actual fault coverage, is numerically much larger. As shown in Table(7.1), it is reduced from 99.36% in the No Cmprs. case to 98.3% under this MISR-based compression.

The ODM column contains the results for the case where the ODM scheme is adopted. The ODM scheme adopted is realized by a limited size Modifier and an Improved Compressor with One's count, added to the standard BIST case. The fault coverage, in this case, is specifically calculated for a Modifier block consisting of  $p = 50$  cubes. The results in Table(7.1) demonstrate that as a result of using such a size of Modifier, the fault coverages reduced under the standard BIST case are totally recovered with the ODM scheme. These improvements in fault coverage reveal the extreme effectiveness of the ODM scheme in eliminating the error masking effect. Hence, with such improvements in fault coverage and particularly for high quality test requirements, the cost of ODM implementation is considered justifiable.

Since the Modifier block is usually of a limited size because of area overhead constraints, the modifier string  $s$  generated is generally not equal to  $q$ . Therefore,  $q' \neq 0$ , and the error masking is not be entirely eliminated, i.e.  $pr \neq 0$ . However, the ODM scheme achieves its aim if this probability  $pr$  is reduced to such a level that it does not affect the fault coverage. For example, the above mentioned modifier size used with the experiments whose results are shown in Table(7.1) does not provide total elimination of the error masking probability. However, it provides such a level of reduction that fault coverages obtained in the No Cmpres. case are recovered. For instance, in mc880, the error masking probability under the ODM scheme is  $pr = 2^{-817}$ , based on the average weight of  $q'$  after modification; whereas, the fault simulation results in Table(7.1) show that the network's fault coverage is brought back to its No Cmpres. case value of 99.36%. Hence, the error masking in the ODM case does not affect the fault coverage.

It can be stated that when the coverage obtained without compression is desired, the reduction in deception volume or error masking probability does not necessarily have to be zero. Therefore, in the cases where the test quality is not limited by a hardware constraint, i.e. by a fixed size Modifier block, but instead by the requirement of providing the full No Cmpres. fault coverage, a lower bound for the deception volume and error masking can be determined. Such bounds can be considered as limits to determine the maximum number of cubes needed to achieve a desired deception volume.

In general, the experimental results shown in this section confirm the significant benefits in terms of fault coverage improvements of adopting the ODM scheme. The results also justify the low cost of ODM implementation in terms of additional hardware.

### 7.3 Conclusive Remarks on the ODM Scheme:

It was originally mentioned that the main goal behind designing a BIST scheme

based on output data modification is to optimize the error making problem such that it becomes insignificant, with the addition of only a small area overhead. A detailed description of the ODM scheme which provides a tremendous reduction in error masking probability with a small amount of hardware is given throughout this dissertation. To complete the analysis of the ODM scheme the following conclusive remarks are made:

- The ODM scheme is a general scheme in the sense that it is not limited to particular circuit topologies, nor does it require the addition of extra hardware to the CUT for it to become testable.
- The implementation of the ODM scheme does not require any details about the structure of the CUT. Hence, it is structure (topology)-independent. In fact, it can be considered function-dependent since it only needs the functionality of the CUT to determine the functional outputs for the input patterns applied.
- In spite of the fact that the IC technology is under continuous improvements, the ODM scheme will not be limited by these changes, since it is based on functional testing, and thus uses functional signatures. Hence, it is independent of the particular technology involved.
- Some of the BIST schemes intended to improve error coverage (section 3.3) cause an increase in the test duration by augmenting the test length, or by splitting the test set to require multiple signatures, or by increasing the number of shifts in the compressor. However, under the ODM scheme, the final signature is accumulated during the test procedure. Hence, this scheme does not cause any additional delays in the generation of the signature, and therefore in the test duration.
- If the test quality of an existing standard BIST design scheme is unsatisfiable, the ODM scheme can easily be added onto the scheme's existing BIST facilities. This

can be realized by simply adding the *Modifier* and the *Improved Compressor* blocks. The addition of these two blocks can even be done outside the chip if necessary.

- Throughout this dissertation it has been assumed that the *Improved Compressor* adopts the *One's Count*-based compression technique. However, the theory and discussions can easily be extended to other count-based, as well as nonlinear compression techniques besides counting. In fact, exploring several count-based techniques provides a larger set of compression functions to select from, and therefore more chances to obtain lower error masking.
- In the *ODM* scheme, the final content (signature) of the *MISR* (*P/S Compressor*) is not utilized in the output data verification process; This is because the signature in the *LFSR-Counter* used for the same purpose generally holds a very low error masking probability ( $2^{-\text{hundreds}}$  or  $2^{-\text{thousands}}$ ) due to output data modification. However, it is possible to use this *LFSR-Counter* content to further enhance the error coverage without incurring any extra hardware cost. This can be realized by setting the *MISR* to a predetermined regular final state [McAnney 86].
- The self-test facilities under the *ODM* scheme can either be entirely added to the original circuit, or formed by reconfiguring some of the existing memory elements in the circuit, in particular to realize the *IPG*, the *P/S Compressor* and the *Improved Compressor*.
- Since the design of each self-test block (section 6.1) is composed of identical cells with uniform interconnections along one dimension, i.e. regular layout features, and since the procedure for the circuit-specific implementation (section 6.2) has the capability of being automated, it would be very realistic and promising to consider developing a CAD tool to provide an automated implementation of the

ODM scheme. Such tool could be embedded in the design for testability tools in a total design methodology where BIST structures are inserted during the early stages of a chip design.

- The ODM scheme has been analytically proven to provide tremendous reduction in deception volume and therefore in error masking probability for any average circuit.
- With the ODM scheme, chip designers are provided with a trade-off for adding more circuitry to achieve a higher assurance that masking will not occur. This can be realized by utilizing a whole range of trade-off curves which enables one to decide the optimal solution between the desired reduction and the available (possible) silicon area overhead.

#### 7.4 BIST schemes adopting the ODM Approach:

Accounts of several attempts suggesting BIST schemes to improve the error coverage recently appeared based on our ODM scheme in the literature. A few remarks on those follows.

One attempt [Hurst 87] uses the ODM approach with some changes in its BIST realization. One of these changes is in the input pattern generation which is limited to exhaustive testing (known to yield long test durations). The other change is in its P/S Compressor, which consists of an Exclusive-OR tree and not an MISR (section 6.1.1.2 shows that the MISR is preferable for this particular task). The most important characteristic of this scheme is that it intends to generate a modifier string  $s$ , which provides a 1 (in the counter) as the final signature. However, the extra hardware cost



for a Modifier block that generates such an exact string (with only one bit difference), would clearly be very expensive.

Another attempt [Li 87] uses the entire ODM scheme with a change in its P/S Compressor structure, which in this case is circuit-specific. More precisely, the P/S Compressor in [Li 87] is designed according to the output data matrix of the CUT, and is composed of a combination of gates selected from a set of three combinational gates. To have a circuit-specific P/S Compressor is, in fact, intended to improve the error coverage. Such P/S Compressor might be helpful if the original non circuit-specific P/S Compressor (MISR) happens to have a considerable error masking probability. However, since this probability for the MISR is  $pr = 2^{-l}$ , and since  $l$  is usually in thousands or millions, then, as mentioned in section 6.1.1.2, the error masking probability for the MISR is already an extremely small number. Therefore, there is no strong need to reduce the error masking at this stage. As it was previously stated, the aim of the ODM scheme is to improve the error coverage of a BISTed design as well. However, the reduction in error masking probability provided by the ODM scheme is from  $2^{-k}$  to  $2^{-\text{thousands}}$ , where  $k$  is the size of the MISR and usually is not more than a two digit number.

In other BIST schemes, e.g. those proposed in [Saxena 85] and [Robinson 87] the suggestion is to simply add the ODM approach to their self-test facilities to improve the error coverage. Even though the optimization of error masking obtained by adopting the ODM scheme makes it a profitable scheme, ODM could be utilized further for other applications. Some of these further possibilities are briefly mentioned in the next section.

## 7.5 Further Possibilities Under ODM:

The ODM scheme as a BIST design, as well as the different methods to easily gen-

erate sequences presented in the previous chapters can be useful for other applications. This section briefly mentions some of these possibilities.

For example, the ODM scheme can be extremely useful in a situation wherein the input patterns applied in a BIST design are deterministically obtained for a 100% fault coverage [Patel 84]. However, when the resulting output is compressed in an MISR, the fault coverage could go down to a lower value depending on the order in which the set of input patterns are applied. The proposed ODM scheme can improve this coverage with a very small amount of additional overhead.

Other than in this specific application, in general, the different methods developed for generating a sequence out of a large number of possible sequences (of length  $l$ ), by using small amounts of hardware ( $O(\log(l))$  gates) [Zorian 84] [Zorian 86a] [Agarwal 87], can also be adopted in various applications other than the ODM scheme.

However, the greatest potential benefit that the ODM scheme holds for further possibilities lies in its proposed structure. In general terms, ODM's structure can be thought of as a structure that tremendously simplifies the original circuit under consideration at a very low cost. To be more specific, if the ODM model without the final Counter block is considered and is analyzed from a general point of view, then it can be seen that this model transfers the original CUT from a general multi-output circuit into a functionality simplified single-output circuit. More importantly, the output sequence  $q'$  generated may be designed to hold a characteristic specifically desired for a particular application under consideration. In fact, the Modifier block can be programmed according to the desired sequence  $q'$ . In the application shown throughout this dissertation, i.e. using the ODM scheme to optimize error masking, the desired characteristic of the sequence  $q'$  is the minimal possible number of ones, i.e. weight. Other such characteristics could be sought in other applications. Hence, in general, the proposed ODM

structure is a non-expensive way to simplify a general multi-output circuit to a single output circuit that generates a special output sequence  $q'$ . This feature can be exploited for various applications. One example is diagnosing faults in BISTed circuits, which is still an open problem [McAnney '87]. This could possibly be realized by using some kind of on-line comparison between the special output sequence  $q'$  and its expected value.

## References

[Agarwal 81] Agarwal, V.K. and Cerny, E., "Store and Generate Built-In-Testing Approach", Proc. 11th Int'l Symp. on Fault-Tolerant Computing Systems (FTCS-11), pp. 35-40, June 1981.

[Agarwal 83] Agarwal, V.K., "Increased Effectiveness of Built-In-Testing by Output Data Modification", Proc. 13th Int'l Symp. on Fault-Tolerant Computing Systems (FTCS-13), pp 227-234, June 1983.

[Agarwal 87] Agarwal, V.K. and Zorian, Y., "An Introduction to an Output Data Modification Scheme", a chapter in "Developments in Integrated Circuit Testing", edited by Miller, D.M., Academic Press, pp. 219-256, Fall 1987.

[Agrawal 75] Agrawal, P. and Agrawal, V.D., "Probabilistic Analysis of Random Test Generation Method for Irredundant Combinational Logic Networks", IEEE Trans. on Computers, C-24, 691-695, 1975.

[Agrawal 82] Agrawal, V.D. and Mercer, M.R., "Testability Measures- What Do They Tell Us?", Proc. IEEE Int'l Test Conference, pp. 391-396, 1982.

[Akers 77] Akers, S.B., "Partitioning for Testability", J. Design Automation and Fault-Tolerant computing, Vol. 1, No. 2, February 1977.

[Akers 85] Akers, S.B., "On The Use of Linear Sums in Exhaustive Testing", Proc. of FTCS-15, pp 148-153, June 1985.

[Ando 80] Ando, H., "Testing VLSI with random access Scan", Proc. Compcon, pp. 50-52, February 1980.

[Archambeau 84] Archambeau, E.C., and McCluskey, E.J., "Fault Coverage of Pseudo-Exhaustive Testing", Proc. of FTCS-14, pp. 141-145, June 1984.

[Armstrong 72] Armstrong, D.B., "A Deductive Method for Simulating Faults in Logic Circuits", IEEE Trans. on Computers, Vol. C-21, pp. 464-471, May 1972.

[Bardell 81] Bardell, P.H. and McAnney, W.H., "A View from the Trenches: Production Testing of a Family of VLSI Multichip Modules", Proc. 11th Int'l Symp. Fault-Tolerant Computing Systems (FTCS-11), pp. 281-283, June 1981.

[Bardell 82] Bardell, P.H., and McAnney, W.H., "Self-Testing of Multichip Logic Modules", IEEE Int'l Test Conference (ITC), pp. 200-204, November 1982.

[Barzilai 81] Barzilai, Z., Savir, J., Markowsky, G. and Smith M.G., "VLSI Self-Testing Based on Syndrome Techniques", Proc. IEEE Int'l Test Conference 81 (ITC), pp. 102-109, 1981; also in "The Weighted Syndrome Sums Approach to VLSI Testing", IEEE Trans. on Computers, Vol. C-30, No. 12, pp. 996-1000, December 1981.

[Barzilai 83] Barzilai, Z., Coppersmith, D. and Rosenberg A., "Exhaustive Bit Pattern Generation in Discontiguous Positions with Applications to VLSI Testing", IEEE Trans. on Computers, Vol. C-32, No. 2, pp. 190-194, February 1983.

[Benowitz 75] Benowitz, N., Calhoun, D.F., Alderson, G.E., Bauer, J.E. and Joeckel, C.T., "An Advanced Fault Isolation System for Digital Logic", IEEE Trans. on Computers, Vol. C-24, No. 5, pp. 489-497, May 1975.

[Bhattacharya 87] Bhattacharya, B.B. and Seth, S.C., "On the Reconvergent Structure of Combinational Circuits with Applications to Compact Testing", Proc. 17th Int'l Symp. on Fault-Tolerant Computing Systems (FTCS-17), pp. 264-269, July 1987.

[Bhavsar 84] Bhavsar, D.K., and Krishnamurthy, B., "Can we Estimate Fault Escape in Self-Testing by Polynomial Division?" Proc. of IEEE Int'l Test Conference 84, pp. 134-139, Philadelphia, PA, October 1984.

[Bhavsar 85] Bhavsar, D.K., "Concatenable Polydividers", Proc. of IEEE Int'l Test Conference 85, pp. 88-93, Philadelphia, PA, October 1985.

[Bose 82] Bose, A.K. et al., "A Fault Simulator for MOS LSI Circuits", Proc. of 19th Design Automation Conference (DAC-19), pp. 400-409, June 1982.

[Bottorff 77] Bottorff, P.S., France, R.E., Garges, N.H. and Orosz, E.J., "Test Generation for Large Logic Networks", Proc. of 14th Design Automation Conference (DAC-14), pp. 479-485, June 1977.

[Bozorgui-Nesbat 80] Bozorgui-Nesbat, S. and McCluskey, E.J., "Structured Design For Testability To Eliminate Test Pattern Generation", Proc. Int'l Symp. on Fault-Tolerant Computing Systems (FTCS-10), pp. 158-163, June 1980.

[Brayton 84] Brayton, R.K. et al., "Logic Minimization Algorithms for VLSI synthesis", Kluwer Academic Publishers, Boston, 1984.

[Breuer 76] Breuer, M.A., "Diagnosis and Reliable Design of Digital Systems", Computer Science Press, Woodland Hills, Cal., 1976.

[Brglez 84] Brglez, F., Pownall, P. and Hum, R., "Application of Testability Analysis: From ATPG to Critical Delay Path Tracing", IEEE Int'l Test Conference, pp. 705-712, October 1984.

[Brglez 85] Brglez, F., Pownall, P. and Hum, R., "Accelerated ATPG and Fault Grading, via Testability Analysis", Proc. of Int'l Symp. on Circuits and Systems (IS-CAS), pp. 695-698, Kyoto, Japan, 1985.

[Carter 79] Carter, J.L. and Wegman, M.N., "Universal Classes of Hash Functions", Journal of Computer and Systems Sciences, Vol. 18, pp. 143-154, April 1979.

[Carter 82a] Carter, J.L., "The Theory of Signature Testing for VLSI", 14-th ACM Symposium on Theory of Computing, pp. 66-76, May 1982.

[Carter 82b] Carter, W.C., "The Ubiquitous Parity Bit", Proc. Int'l Symp. on Fault-Tolerant Computing Systems (FTCS-12), pp. 289-296, June 1982.

[Carter 82c] Carter, W.C., "Signature Testing with Guaranteed Bounds for Fault Coverage", Proc. IEEE Int'l Test Conference (ITC 82), pp. 75-82, November 1982.

[Chan 77] Chan, A.Y., "Easy-to-Use Signature Analyzer Accurately Troubleshoots Complex Logic Circuits", Hewlett-Packard Journal, pp. 9-14, May 1977.

[Chang 70] Chang, H.Y., Manning, E.G. and Metze, G., "Fault Diagnosis of Digital Systems", New York: Wiley-Interscience, 1970.

[Chin 84] Chin, C. and McCluskey, E.J., "Weighted Pattern Generation for Built-In Self-Test", Stanford Univ., Center for Reliable Computing, Tech. Rep. 84-249. August 1984.

[Chin 87] Chin, C. and McCluskey, E.J., "Test Length for Pseudo-Random Testing", IEEE Trans. on Computers, Vol. C-36, No. 2, pp. 252-256, February 1987.

[Dagenais 85] Dagenais, M.R., Agarwal, V.K., and Rumin, N.C., "The McBoole Logic Minimizer", Proc. of 22nd Design Automation Conference (DAC-22), pp. 667-673, June 1985.

[Danniels 85] Danniels, R.G. and Bruce, W.C., "Built-In Self-Test Trends in Motorola Microprocessors", IEEE Design and Test of Computers, pp. 64-71, 1985.

[DasGupta 78] DasGupta, S., Eichelberger, E.B. and Williams, T.W., "LSI Chip Design for Testability", Proc. IEEE Solid-State Circuits Conference, pp. 216-217, February 1978.

[DasGupta 82] DasGupta, S., Goel, P., Walter, R.G. and Williams, T.W., "A Variation of LSSD and its Implication on Design and Test Pattern Generation in VLSI", Proc. IEEE Int'l Test Conference, pp. 63-66, 1982.

[DasGupta 84] DasGupta, S., Graf, M.C., Rasmussen, R.A., Walter, R.G. and Williams, T.W., "Chip Partitioning Aid: A Design Technique for Partitionability and Testability in VLSI", Proc. 21st Design Automation Conference, pp. 203-208, 1984.

[David 76] David, R. and Blanchet, G., "About random fault detection in combinational networks", IEEE Trans. on Computers, C-25, No. 6, pp. 659-664, June 1976.

[David 78] David R., "Feedback Shift Register Testing", Proc. Int'l Symp. on Fault-Tolerant Computing Systems (FTCS-8), pp. 103-107, June 1978; or "Testing by Feedback Shift Register", IEEE Trans. on Computers, Vol. C-26, No. 7, pp. 668-673, July 1980.

[David 86] David, R., "Signature Analysis for Multiple-Output Circuits", IEEE Trans. on Computers, Vol. C-35, No. 9, pp. 830-837, September 1986.

[Davidson 81] Davidson, Proc. IEEE Int'l Test Conference, pp. 15-20, November 1981.

[Dietmeyer 79] Dietmeyer, L., *Logic Design of Digital Systems*, Allyn and Bacon, Boston Mass., March 1979.

[Eichelberger 77] Eichelberger E.B. and Williams T.W. "A Logic Design Structure for LSI Testability", Proc. 14-th Design Automation Conference, June 1977.

[Eichelberger 83] Eichelberger, E.M. and Lindbloom, E., "Random Pattern Coverage Enhancement for LSSD Logic Self-Test", IBM J. Res. Develop., Vol. 27, pp. 265-272, May 1983.

[Eiki 80] Eiki, H., Inagaki, K. and Yajima, S., "Autonomous Testing and its Application to Testable Design of Logic Circuits", Proc. 10th Int'l Symp. Fault-Tolerant Computing Systems (FTCS-10), pp. 173-178, Kyoto, Japan, October 1980.

[Elspas 59] Elspas, B., "The Theory of Autonomous Linear Sequential Networks", IRE Trans. on Circuit Theory, Vol. CT-6, No. 1, pp. 45-60, March 1959.

[Fasang 80] Fasang, P.P., "BIDCO, Built-In Digital Circuit Observer", Proc. IEEE Int'l Test Conference (ITC), pp. 261-266, November 1980.

[Frohwerk 77] Frohwerk R.A., "Signature Analysis: A New Digital Field Service Method", Hewlett Packard Journal, pp. 2-8, May 1977.

[Fujiwara 78] Fujiwara, H., and Kinoshita, K., "Testing logic circuits with compressed data", in Proc. FTCS-8, pp.108-113, June 1978.

[Fujiwara 82] Fujiwara, H., and Toida, S., "The Complexity of Fault Detection Problems for Combinational Logic Circuits", IEEE Trans. on Computers, Vol. C-31, No. 6, pp. 555-560, June 1982.

[Fujiwara 83] Fujiwara, H. and Shimono, T., "On the Acceleration of Test Generation Algorithms", IEEE Trans. on Computers, Vol. C-32, No. 12, pp. 1137-1144, December 1983.

[Funatsu 75] Funatsu, S., Wakatsuki, N. and Arima T., "Test Generation Systems in Japan", Proc. 12th Design Automation Symp., pp. 114-122, June 1975.

[Galiay 80] Galiay, J., Crouzet, Y. and Vergniault, M., "Physical Versus Logical Fault Models MOS LSI Circuits: Impact on Their Testability", IEEE Trans. on Computers, Vol. C-29, No. 6, pp. 527-531, June 1980.

[Gelsinger 86] Gelsinger, P., "Built In Self Test of the 80386", Proc. of Int'l Conference on Computer Design (ICCD), pp. 169-173, October 1986.

[Goel 81] Goel, P., "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits", IEEE Trans. on Computers, Vol C-30, No. 3, pp. 215-222, March 1981.

[Golan 86] Golan, P., Nacak, O. and Hilavcka, J., "Pseudoexhaustive Test Pattern Generator with Enhanced Fault Coverage", Proc. FTCS-16, pp.404-409, July 1985, Vienna.

[Goldstein 79] Goldstein, L.H., "Controllability / Observability Analysis of Digital Circuits", IEEE Trans. on Circuits and Systems, Vol. CAS-26, No. 9, pp. 685-693, September 1979.

[Golomb 82] Golomb, S.W., "Shift Register Sequences", revised edition Laguna Hills, CA: Aegean Park, 1982.

[Gschwind 75] Gschwind, H.W. and McCluskey, E.J., "Design of Digital Computers", New York: Springer, 1975.

[Hassan 83] Hassan, S.Z., Lu, D.J. and McCluskey, E.J., "Parallel Signature Analyzers - Detection Capability and Extensions", Proc. Compcon Spring, pp. 440-445, 1983.

[Hassan 84a] Hassan, S.Z., and McCluskey, E.J., "Increased Fault Coverage through Multiple Signatures", Proc. of 14th Int'l Symposium on Fault Tolerant Computing Systems (FTCS-14), pp. 354-359, June 1984.

[Hassan 84b] Hassan, S.Z. and McCluskey, E.J., "Increasing Effective Fault Coverage of Parallel Signatures Analyzers", CRC TR., Center for Reliable Computing, Computer Systems Laboratory, Stanford, California, 1984.

[Hayes 74] Hayes, J.P. and Friedman, A.D., "Test Point Placement to Logic Block Observation Techniques", Proc. IEEE Test Conference, pp. 37-41, June 1973.

[Hayes 76a] Hayes, J.P., "Check Sum Methods for Test Data Compression", Journal of Design Automation and Fault-Tolerant Computing, pp. 3-17, June 1976.

[Hayes 76b] Hayes, J.P., "Transition Count Testing of Combinational Logic Circuits", IEEE Trans. on Computers, Vol. C-25, No. 6, pp. 613-620, June 1976; or Proc. 5-th Int'l Symp. on Fault-Tolerant Computing Systems (FTCS-5), pp. 215-219, Paris, June 1975.

[Hayes 78] Hayes, J.P., "Generation of Optimal Transition Count Testing", IEEE Trans. on Computers, Vol. C-27, pp. 36-41, 1978.

[Hayes 82] Hayes, J.P., "A Fault Simulation Methodology for VLSI", Proc. of 19th Design Automation Conference (DAC-19), pp. 393-399, June 1982.

[Heckelman 81] Heckelman, R.W. and Bhavsar, D.K., "Self-Testing VLSI", Proc. IEEE Solid State Circuits Conference, pp. 174-175, February 1981.

[Hlawiczka 81] Hlawiczka, A. and Kubica, H., "The Method of Increasing the Probability of Detecting Errors for Signature analysis", Proc. of 4th Int'l Conference on Fault Tolerant Systems and Diagnostics (FTSD-4), pp. 273-277, Burno, September 1981.

[Hlawiczka 86] Hlawiczka, A., "Parallel Multisignature Analysis of Faults in the Multi-Output Digital System", Proc. of 16th Int'l Symp. on Fault Tolerant Computing Systems (FTCS-16), pp. 398-403, Vienna, July 1986.

[Hsiao 77] Hsiao, M.Y., Patel, A.M., and Pradhan, D.K., "Store Address Generator with On-Line Fault Detection Capability", IEEE Trans. on Computers, Vol. C-26, No. 11, pp. 1144-1147, November 1977.

[Hurst 85] Hurst, S.L., Miller, D.M. and Muzio, J.C., "Spectral Techniques in Digital Logic", Academic Press, New York 1985.

[Ibarra 75] Ibarra, O., H., and Sahni, S.K., "Polynomially Complete Fault Detection Problems", IEEE Trans. on Computers, Vol. C-24, No. 3, pp. 242-249, March 1975.

[Ivanov 87] Ivanov, A. and Agarwal, V.K., "On a Fast Method to Monitor the Behavior of Signature Analysis Registers", Proc. IEEE Int'l Test Conference (ITC 87), pp. 645-655, September 1987.



[Jain 84] Jain, S.K. and Agrawal, V.D., "STAFAN: An Alternative to Fault Simulation", Proc. of 21st Design Automation Conference, pp. 18-23, June 1984.

[Komonytsky 82] Komonytsky, D., "LSI Self-Test Using Level Sensitive Scan Design and Signature Analysis", Proc. IEEE Int'l Test Conference, pp 414-424, November 1982.

[Komonytsky 83] Komonytsky, D., "Synthesis of Techniques Creates Complete System Self-Test", Electronics, pp. 110-115, March 1983.

[Konemann 79] Konemann, B., Mucha, and Zwiehoff, G., "Built-In logic Block Observation Technique", Proc. IEEE Test Conference, pp. 37-41, October 1979.

[Konemann 80] Konemann, B., Mucha, J. and Zwiehoff, G., "Built-In Test for Complex Digital Integrated Circuits", IEEE Journal Solid-State Circuits, Vol. SC-15, No. 3, pp. 315-318, June 1980.

[Kuban 83] Kuban, J. and Bruce, B., "The MC6804P2 Built-In Self-Test", Proc. IEEE Int'l Test Conference (ITC), pp. 295-300, 1983.

[Levendel 81] Levendel, Y.H. and Menon P.R., "Fault Simulation Methods - Extensions and Comparison", Bell Systems Tech. Journal, Vol. 60, No. 9, pp. 2235-2258, November 1981.

[Li 87] Li, Y.K. and Robinson, J.P., "Space Compression Methods with Output Data Modification", IEEE Trans. on Computer Aided Design, Vol. CAD-6, No. 2, pp. 290-294, March 1987.

[Losq 78] Losq, J., "Efficiency of Random Compact Testing", IEEE Trans. on Computers, Vol. C-27, No. 6, pp. 516-525 June 1978.

[Malaiya 84] Malaiya, Y.K. and Yang, S., "The Coverage Problem for Random testing", Proc. IEEE Int'l Test Conference, pp. 237-245, November 1984.

[Mangir 83] Mangir, T.E., "Design for Testability - An Integrated Approach to VLSI Testing", Proc. Int'l Conference on Computer-Aided Design (ICCAD 83), pp. 68-70, September 1983.

[Mann 80] Mann, W.R., "Microelectronic Device Test Strategies - A Manufacturer's Approach", Proc. IEEE Int'l Test Conference, pp.195-202, November 1980.

[Markowsky 81] Markowsky, G., "Syndrome testability Can Be Achieved by Circuit Modification", IEEE Trans. on Computers, Vol. C-30, pp. 606-609, 1981.

[McAnney 86] McAnney, W.H., and Savir J., "Built-In Checking of the Correct Self-Test Signature", Proc. IEEE Int'l Test Conference, pp. 54-58, Sept. 1986.

[McAnney 87] McAnney, W.H. and Savir, J., "There is Information in Faulty Signatures", proc. of IEEE Int'l Test Conference, pp. 630-636, September 1987.

[McCluskey 81] McCluskey, E.J. and Bozorgui-Nesbat, "Design for Autonomous Test", IEEE Trans. on Circuits and Systems, Vol. CAS-28, No. 11, pp. 1070-1079, November 1981.

[McCluskey 82] McCluskey, E.J., "*Built-In Verification Test*", Proc. of IEEE ITC 82, pp. 183-190, Philadelphia, PA, November 1982.

[McCluskey 84a] McCluskey, E.J., "*Verification Testing - A Pseudo-Exhaustive Test Technique*", IEEE Trans. on Computers, Vol. C-33, No. 6, pp. 541-546, June 1984.

[McCluskey 84b] McCluskey, E.J., "*A Survey of Design for Testability Scan Techniques*", VLSI Design, Vol. V, No. 12, pp. 38-61, December 1984.

[McCluskey 85a] McCluskey, E.J., "*Built-In Self-Test Techniques*", IEEE Design and Test, Vol. 2, No. 2, pp. 21-28, April 1985.

[McCluskey 85b] McCluskey, E.J., "*Built-In Self-Test Structures*", IEEE Design and Test, Vol. 2, No. 2, pp. 29-36, April 1985.

[McCluskey 86] McCluskey, E.J., "*Logic Design Principles with Emphasis on Testable Semicustom Circuits*", Prentice-Hall, Englewood Cliffs, N.J. 1986.

[McCluskey 87] McCluskey, E.J., Makar, S., Mourad, D., and Wagner, K.D., "*Probability Models for Pseudo-Random Test Sequences*", Proc. of IEEE Int'l Test Conference (ITC 87), pp. 471-479, September 1987.

[Mead 80] Mead, C., and Conway, L., "*Introduction to VLSI Systems*", Addison-Wesley Publishing Company, 1980.

[Mei 74] Mei, K.C.Y., "*Bridging and stuck-at faults*", IEEE Trans. on Computers, Vol. C-23, No. 7, pp. 720-727, July 1974.

[Mileto 64] Mileto, M. and Putzolu, G., "*Average Values of Quantities Appearing in Boolean Function Minimization*", IEEE Trans. on Computers, pp. 87-92, April 1964.

[Miller 84] Miller, D.M. and Muzio, J.C., "*Spectral Fault Signatures for Single Stuck-at Faults in Combinational Networks*", IEEE Trans. on Computers, Vol. C-33, No. 8, pp. 765-769, August 1984.

[Mucha 81] Mucha, J., "*Hardware Techniques for Testing VLSI Circuits Based on Built-In Test*", Proc. COMPCON 81, pp. 336-369, February 1981.

[Muehldorf 81] Muehldorf, E.I. and Savkar, A.D., "*LSI Logic Testing- An Overview*", IEEE Trans. on Computers, Vol. C-30, No. 1, pp. 1-16, January 1981.

[Muzio 83] Muzio, J.C. and Miller, D.M., "*Spectral Fault Signatures for Internally Unate Combinational Networks*", IEEE Trans. on Computers, Vol. C-32, pp. 1058-1062, 1983.

[Myers 83] Myers, M.A., "*An Analysis of the Cost and Quality Impact of LSI-VLSI Technology on PCB Test Strategies*", Proc. IEEE International Test Conference, pp. 382-395, October 1983.

[Newkirk 1983] Newkirk, J. and Mathews, R., "*The VLSI Designer's Library*", Addison-Wesley Publishing Company, Inc., Advanced Book Program, Reading, Mass. 1983.

[Ohletz 87] Ohletz, M.J., Williams, T.W. and Mucha, J.P., "Overhead in Scan and Self-Testing Designs", Proc. of IEEE Int'l Test Conference (ITC 87), pp 460-470, September 1987.

[Parker 76] Parker, K.P., "Compact Testing: Testing with Compressed Data", Proc. 6th Int'l Symp. on Fault-Tolerant Computing Systems (FTCS-6), pp. 93-98, June 1976.

[Parker 79] Parker, K.P., "Software Simulator Speeds Digital Board Test Generation", Hewlett-Packard Journal, pp. 13-19, March 1979.

[Patel 84] Patel, J.H., and Abraham, J.A., "Design of Test Pattern Generators for Built-In Test", Proc. of IEEE Int'l Test Conference, pp. 315-319, October 1984.

[Perkins 80] Perkins, C.C. et al., "Design for In-Situ Chip Testing with a Compact Tester", Proc. Int'l Test Conference, pp. 29-41, November 1980.

[Peterson 75] Peterson, W.W., and Weldon, E.J.Jr., *Error Correcting Codes*, 2nd edition, MIT Press, Cambridge Mass. and London, December 1975.

[Reddy 77] Reddy, S.M., "A Note on Testing Logic Circuit Transition Counting", IEEE Trans. on Computers, Vol. C-26, pp. 313-314, 1977.

[Reddy 85] Reddy, S.M., Saluja, K.K. and Karpovsky, M., "A Data Compression Technique for Built-In Self-Test", Proc. Int'l Symp. on Fault-Tolerant Computing Systems (FTCS 15), pp. 294-299, July 1985.

[Resnick 83] Resnick, D.R., "Testability and Maintainability with a New 6K Gate Array", VLSI Design, pp. 34-38, March/April 1983.

[Robinson 87] Robinson, J.P. and Saxena, N.R., "A Unified View of Test Compression Methods", IEEE Trans. on Computers, Vol. C-36, No. 1, pp. 94-99, January 1987.

[Roth 66] Roth, J.P., "Diagnosis of Automata Failures: A Calculus and a Method", IBM J. Res. Develop., Vol. 10, pp. 278-291, July 1966.

[Saluja 83] Saluja, K.K. and Karpovsky, M., "Test Compression Hardware through Data Compression in Space and Time", Proc. IEEE Int'l Test Conference (ITC 83), pp. 83-88, 1983.

[Saluja 87] Saluja, K.K., Sng, S.H. and Kinoshita, K., "Built-In Self-Testing RAM: A Practical Alternative", IEEE Design and Test of Computers, Vol. 4, No. 1, pp. 42-51, February 1987.

[Savir 80] Savir, J.E., "Syndrome Testable Design of Combinational Circuits", IEEE Trans. on Computers, Vol. C-29, No. 6, pp. 442-550, June 1980.

[Savir 81] Savir, J.E., "Syndrome Testable of Syndrome Untestable Combinational Circuits", IEEE Trans. on Computers, Vol. C-30, No. 8, pp. 606-609, August 1981.

[Savir 83a] Savir, J.E., Detlow, G. and Bardell, P.H., "Random Pattern Testability", Proc. Int'l Symp. on Fault-Tolerant Computing Systems (FTCS-13), pp. 80-89, 1983; or IEEE Trans. on Computers, Vol. C-33, No. 1, pp.79-90, January 1984.

[Savir 83b] Savir, J.E. and Bardell, P.H., "On Random Pattern Test Length", Proc. Int'l Test Conference, pp. 95-106, 1983; or IEEE Trans. on Computers, Vol. C-33, No. 6, pp. 467-474, June 1984.

[Savir 85] Savir, J.E. and McAnney, W.H., "On The Masking Probability with One's Count and Transition Count", Proc. Int'l Conference on Computer Aided Design (ICCAD 85), pp. 111-113, 1985.

[Saxena 85] Saxena, N.R., and Robinson, J.P., "Accumulator Compression Testing", FTCS-15, pp.300-305, June 1985; or IEEE Trans. on Computers, Vol. C-35, pp. 317-321, 1986.

[Saxena 87] Saxena, N.R. and Robinson, J.P., "Syndrome and Transition Count are Uncorrelated", will appear in IEEE Trans. on Information Theory.

[Schnurmann 75] Schnurmann, H.D., Lindbloom, E. and Carpenter, R.G., "The Weighted Random Test Patterns Generation", IEEE Trans. on Computers, Vol. C-24, No. 7, pp. 695-700, July 1975.

[Sedmak 79] Sedmak, R.M., "Design for Self-Verification: An Approach for Dealing with Testability Problems in VLSI-Based Designs", Proc. IEEE Test Conference, pp.112-124, October 1979.

[Sedmak 80] Sedmak, R.M., "Implementation Techniques for Self-Verification", Proc. IEEE Int'l Test Conference (ITC), pp. 267-278, November 1980.

[Segers 81] Segers, M.T.M., "A Self-Test Method for Digital Circuits", Proc. IEEE Int'l Test Conference (ITC 81), pp. 79-85, Oct. 1981.

[Seshu 65] Seshu, S., "On an Improved Diagnosis Program", IEEE Trans. on Electronic Computers, Vol. EC-12, No. 1, pp. 76-79, February 1965.

[Seth 85] Seth, S.C., Pan, L. and Agrawal, V.D., "Predict- Probabilistic Estimation of Digital Circuit Testability", Proc. 15th Int'l Symp. on Fault-Tolerant Computing Systems (FTCS-15), pp. 220-225, June 1985.

[Shedletsky 75] Shedletsky, J.J. and McCluskey, E.J., "The Error Latency of a Fault in a Combinational Digital circuit", Proc. 5th Int'l Symp. on Fault-Tolerant Computing Systems (FTCS-5), pp. 210-214, June 1975.

[Shedletsky 77] Shedletsky, J.J., "Random Testing: Practicality versus Verified Effectiveness", Proc. Int'l Symp. on Fault-Tolerant Computing Systems (FTCS-7), pp. 175-179, 1977.

[Smith 80] Smith, J.E., "Measures of Effectiveness of Fault Signature Analysis", IEEE Trans. on Comp., C-29, pp. 510-514, June 1980.

[Sridhar 82] Sridhar, T., Ho, D.S., Powell, T.J., and Thatte S.M. "Analysis and Simulation of Parallel Signature Analyzers", Proc. IEEE ITC, pp.656-661, Oct. 1982.

[Stewart 77] Stewart, J.H., "Future Testing of Large LSI Circuit Cards", Proc. Semiconductor Test Symp., pp. 6-15, October 1977.

[Sun 84] Sun, Z. and Wang, L.-T., "Self-Testing of Embedded RAMs", Proc. IEEE Int'l Test Conference (ITC), pp. 148-156, October 1984.

[Susskind 81] Susskind, A.K., "Testing by Verifying Walsh Coefficients", Proc 11-th Int'l Symp. on Fault-Tolerant Computing Systems (FTCS-11), pp. 206-208, 1981.

[Tang 83] Tang, D.T. and Woo, L.S., "Exhaustive Test Pattern Generation with Constant Weight Vectors", IEEE Trans. on Computers, Vol. C-32, No. 12, pp. 1145-1150, December 1983.

[Tang 84] Tang, D.T., and Chen, S.L., "Logic Test Pattern Generation Using Linear Codes", Proc. 13-th Int'l Symp. on Fault-Tolerant Computing Systems (FTCS-13), pp. 222-226, June 1983; or IEEE Trans. on Computers, Vol. C-33, No.6, pp. 845-850, June 1984.

[Timoc 83] Timoc, Proc. IEEE Int'l Test Conference, pp.701-703, 1983.

[Treuer 85] Treuer, R., Fujiwara, H. and Agarwal V.K., "Implementing a Built-In Self-Test PLA Design", IEEE Design and Test of Computers, Vol. 2, No. 2, pp. 37-48, April 1985.

[Tzidon 78] Tzidon, A., Berger, I. and Yoeli, M., "A Practical Approach to Fault Detection in Combinational Networks", IEEE Trans. in Computers, Vol. C-27, No. 10, pp. 968-971, October 1978.

[Ulrich 74] Ulrich, E.G. and Baker, T., "Concurrent Simulation of Nearly Identical Digital Networks", Computer, Vol. 7, No. 4, pp. 39-44, April 1974.

[Vasanthavada 85] Vasanthavada, N. and Marinos, P.N., "An Operationally Efficient Scheme for Exhaustive Test Pattern Generation Using Linear Codes", Proc. IEEE Int'l Test Conference 85, pp. 476-482, Philadelphia, PA, November 1985.

[Wadsack 78] Wadsack, R.L., "Fault Modeling and-Logical Simulation of CMOS and MOS integrated circuits", Bell System Technical Journal, Vol. 57, pp. 1449-1474, May 1978.

[Wagner 83] Wagner, K.D., "Design for Testability in the Amdahl 580", COMP-CON, 1983.

[Wagner 87] Wagner, K.D., Chin, C.K. and McCluskey E.J., "Pseudorandom Testing", IEEE Trans. on Computers, Vol. C-36, No.3. pp. 332-343; March 1987.

[Waicukauski 85] Waicukauski, J.A., Eichelberger, E.A., Forlenza, D.O. Lindbloom, E. and McCarthy, T., "Fault Simulation for Structured VLSI", VLSI Systems Design, pp. 20, December 1985.

[Wang 82] Wang L.-T., "Autonomous Linear Feedback Shift Register with On-Line Fault Detection Capability", Proc. 12th Int'l Symp. on Fault-Tolerant Computing Systems (FTCS-12), pp. 311-314, 1982.

[Wang 84] Wang L.-T., and McCluskey E.J., "A New Condensed Linear Feedback Shift Register Design for VLSI/System Testing". Proc. of FTCS-14, pp.360-365, Kissimmee, Florida, June 1984; or "Condensed Linear Feedback Shift Register (LFSR)

*Testing- A Pseudoexhaustive Test Technique*", IEEE Trans. on Computers, C-35, No. 4, pp. 367-370, April 1986.

[Wang 85] Wang, L.T. and McCluskey E.J., "Built-In Self-Test for Random Logic", Proc. of IEEE Int'l Symposium on Circuits and Systems 85 (ISCAS-85), Vol. 3 of 3, pp. 1305-1308, Kyoto, Japan, June 1985.

[Wang 86a] Wang, L.T. and McCluskey, E.J., "Complete Feedback Shift Register Design For Built-In Self-Test", IEEE Int'l Conference on Computer Aided Design (ICCAD 86), pp. 56-59, 1986.

[Wang 86b] Wang, L.T. and McCluskey, E.J., "Circuits for Pseudo-Exhaustive Test Pattern Generation", Proc. of IEEE Int'l Test Conference 86, pp. 25-37, Washington, D.C., September 1986.

[Wang 86c] Wang, L.T. and McCluskey, E.J., "A Hybrid Design of Maximum-Length Sequence Generators", Proc. of IEEE Int'l Test Conference 86, pp. 38-47, Washington, D.C., September 1986.

[Wegman 81] Wegman, M.N. and Carter, J.L., "New Hash Functions and their Use in Authentication and Set Equality", Journal of Computer and Systems Sciences, Vol. 22, pp. 265-279, June 1981.

[Williams 73] Williams, M.J.Y. and Angel, J.B., "Enhancing Testability of Large Scale Integrated Circuits via Test Points and Additional Logic", IEEE Trans. on Computers, Vol. C-22, No. 1, pp. 46-60, January 1973.

[Williams 77] Williams, T.W. and Eichelberger, E.B., "Random Patterns Within a Structured Sequential Logic Design", Proc. Semiconductor Test Symposium, pp. 19-27, 1977.

[Williams 79] Williams, T.W. and Parker, K.P., "Testing Logic Networks and Designing for Testability", Computer, Vol. 12, No. 10, pp. 9-20, October 1979.

[Williams 83] Williams, T.W., and Parker, K.P., "Design for Testability - A Survey", IEEE Trans. on Computers, Vol. C-31, No. 1, pp. 2-15, Jan. 1982; also in Proceedings of IEEE, Vol. 71, No. 1, pp. 98-112, January 1983.

[Williams 84] Williams, T.W., "VLSI Testing", IEEE Computer, pp.126-136, 1984.

[Williams 85] Williams, T.W., "Test Length in a Self-Testing Environment", IEEE Design and Test, pp. 59-63, April 1985.

[Williams 86] Williams, T.W., Daehn W., Gruetzner M., and Starke C.W., "Comparison of Aliasing Errors for Primitive and Non-primitive Polynomials", Proc. of IEEE Int'l Test Conference (ITC 87), pp. 282-288, Sept. 1986; or IEEE Design and Test of Computers, Vol. 4, No. 2, pp. 39-45, April 1987.

[Williams 87] Williams, T.W., Daehn, W., Gruetzner, M. and Starke, C.W., "Aliasing Errors with Primitive and Non-Primitive Polynomials", Proc. of IEEE Int'l Test Conference (ITC 87), pp. 637-644, September 1987.

[Wunderlich 85] Wunderlich, H.-J., "Protest: A Tool for Probabilistic Testability Analysis", Proc. of Design Automation Conference (DAC-22), 1985.

[Wunderlich 87] Wunderlich, H.-J., "Self-Test Using Unequiprobable Random Patterns", Proc. 17th Int'l Symp. on Fault-Tolerant Computing Systems (FTCS-17), pp.258-263, July 1987.

[Zasio 83] Zasio, J.J., "Shifting Away From Probes for Wafer Test", Proc. Compcon Spring 83, pp. 395-398, February 1983.

[Zorian 84] Zorian, Y., and Agarwal, V.K., "Higher Certainty of Error Coverage by Output Data Modification", Proc. IEEE Int'l Test Conference, pp. 140-147, Oct. 1984.

[Zorian 85] Zorian Y., and Agarwal V.K., "A General Scheme to Optimize Error Masking in Built-In Self-Testing", Technical Report No 85-36R, VLSI Design Laboratory, McGill University, Dec. 1985.

[Zorian 86a] Zorian Y., and Agarwal V.K., "Ensured Reduction in Error Masking by Output Data Modification", Proc. of Technical Workshop: New Directions for IC Testing, pp 2-1/2-14, March 1986, Victoria, British Columbia.

[Zorian 86b] Zorian Y., and Agarwal V.K., "A General Scheme to Optimize Error Masking in Built-In Self-Testing", Proc. 16-th Int'l Symp. on Fault-Tolerant Computing systems (FTCS-16), pp. 410-415, Vienna, July 1986.

[Zorian 86c] Zorian Y., and Agarwal V.K., "On Improving the Effectiveness of the Standard BIST Approach", Proceedings of the 6-th International Conference on Custom and Semicustom IC's, Nov. 1986, London, England.