A New Procedure for Weighted Random Built-In Self-Test

Fidel Muradali
Bachelor of Engineering

Department of Electrical Engineering
McGill University

A thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Master of Engineering

© Fidel Muradali March 1990.

Table of Contents

List of Figures

List of Tables

Abstract

Experience has shown that an excessive time penalty can be incurred when testing large scan circuits with a uniform random test pattern generation approach. As a solution to this problem, this work explores the use of weighted random patterns (WRP) to reduce, by orders of magnitude, the test application time in self-testing circuits.

Much work has been done on the off-line development of compact test sets, but a problem which still remains is how to efficiently apply them on-chip. A means of transforming a given test set into a relatively short weighted sequence and pseudorandom sequence, whose cumulative fault coverages approximate that guaranteed by the original test set, is proposed.

The single weight set is formulated using a method which does not explicitly consider the circuit structure. Instead, sufficient circuit information contained in the given test set can be extracted using simulation techniques. This is done by analyzing a random pattern detection profile and isolating the *vectors* which cover faults difficult to detect using random patterns. After extracting the useful bits from these vectors, a weight set characteristic of the corresponding faults is estimated as the ratio of 1's to 0's at each bit (input) position.

The generation scheme is evaluated using five large scannable circuits. A *local* approach to on-chip pattern generation is examined.

Résumé

L'expérimentation démontre que la vérification de circuits complexes avec chaîne de balayage par une génération aléatoire uniforme de patrons de test peut s'avérer très longue. Cette recherche étudie l'utilisation de patrons aléatoires pondérés pour diminuer significativement le temps de test des circuits auto-vérifiants.

Plusieurs recherches sur le développement d'ensemble compact de patrons de test ont été effectuées mais leur utilisation efficace à l'intérieur même des puces demeure un problème. Nous proposons une façon de transformer un ensemble donné de patrons de test en une relativement courte séquence pondérée et une séquence pseudo-aléatoire, dont la couverure cumulative approxime celle de la séquence originale. L'ensemble de poids est calculé selon une méthode qui ne considère pas explicitement la structure du circuit. Il est assumé que suffisament d'information est contenue dans l'ensemble des patrons de test.

Les vecteurs qui couvrent des dfauts difficiles à détecter par des patrons aléatoires sont isolés en analysant un profile de patron aléatoire de détection. Les bits utiles à la détection sont isolés de chacun des vecteurs de test. Un poids correspondant au rapport du nombre de 1 sur le nombre de 0 est attribué à chacune des entrées. Un poids uniforme de .5 est utilisé pour couvrir les autres défauts vérifiables.

Le procédé de génération a été evalué en utilisant cinq gros circuits à chaîne de balayage. Une approche locale est présentée pour la génération interne de patrons de test.

Acknowledgements

I would like to take this opportunity to thank my supervisor Dr. V. K. Agarwal for his guidance throughout my term as a graduate student, and Benoit Nadeau-Dostie for his invaluable insight and practical suggestions which led to a more realistic solution to the problems addressed in this thesis. Also, to Rob Aitken for taking on the undesirable task of proof reading the text, and to Francis Larochelle for his french translation skills. I would especially like to thank Ashish Pancholy for assisting with the post-review corrections in my absence Finally, a "thanks guys/gals" goes out to all the lab members for making studying an enjoyable experience.

This work was supported in part by a grant from Fonds pour la Formation de Chercheurs et L'Aide à la Recherche (FCAR) and test circuits were provided by Bell Northern Research (BNR).

Chapter 1 Introduction

Today it is not uncommon to find VLSI chips containing hundreds of thousands to over a million circuit elements, and with continued refinements to packaging and submicron fabrication technologies, circuit density is expected to increase. Inevitably the question must be asked, "...but does it work?" The field of *testing* endevours to respond to this concern.

In modern circuits, testing accounts for roughly a third of a chip's production cost [Bha89]. In fact, it has been found [Wil83] [Bar87] that the price of testing increases approximately five to ten times per level of packaging, to the point where thousands of dollars are at stake if tests are performed in the field. Thus, one way to reduce long term test costs is to ensure that component tests are as thorough as possible at earlier stages of assembly (e.g. probe and chip levels). Also, if possible, the design of the circuit itself should facilitate decreased test effort and increased test effectiveness. Such linking of the design process with testing has resulted in a knowledge base of design techniques called "Design for Testability" (DFT). An example of this is built-in self-test (BIST) wherein on-chip and/or on-board circuits provide and analyze test data. The technique substantially reduces the dependency on external test units and can thus simplify in-field maintenance.

This thesis demonstrates the use of DFT concepts to develop a new BIST strategy intended for large circuits (hundreds of thousands of gates) with thousands of I/O signals. In general, the amount of input data needed to test a circuit is proportional

to the square of the number of circuit gates [Goe81] As such the storage requirements of the test hardware and test application time also grow quadratically with circuit size Thus as circuit density approaches that of the circuits under consideration, these factors may become bottlenecks which affect the cost of performing a through test

In test generation method proposed, input data is pseudorandomly generated in-circuit to decrease the storage requirements needed. Furthermore, in addition to using a conventional uniform random pattern generation approach [Bar87], the volume of input data is significantly reduced by generating non-uniform, or weighted, random pattern sequences. Experimental results for this scheme record test times orders of magnitude smaller than that of standard uniform random pattern testing

The structure of this thesis follows the breakdown of the test process into two off-line steps :

- Test generation, and
- Test set verification

and 2 practical steps :

- Test pattern application, and
- Test response evaluation

However, the emphasis is placed mainly on the front-end test generation and test application processes.

Chapter 2 introduces the concepts involved in test generation. This is basically the compilation of a set of input stimuli which verifies whether a circuit is defective or not Test set verification is the evaluation of the effectiveness of a given test set to this end This is usually recorded as a measure of the percentage of modeled faults which can be detected This measure is called the *fault coverage* of the test set According to the manner in which the test set is developed verification may be done implicitly (e g ATPG section 2.3.1), or with the use of simulation which is a brute force approach Further discussion of test set verification is not included in this thesis.

As a transition from the off-line steps to the practical issues, DFT techniques pertinent

to the research at hand are summarized in chapter 3. Chapter 4 then describes some of the hardware used to physically apply test input to the circuit under test (CUT), and analyse the CUT's response to this data. An introduction to on-chip testing is also given in this chapter.

Chapter 5 is a brief overview of some known algorithms and circuits used to generate weighted random patterns. Chapter 6 contains a detailed examination of the proposed weighted random BIST strategy, and comments on the computational overhead required and the testability of the circuits designed. Supplementary details concerning the experiments performed are contained in the appendices.

Chapter 2 An Overview of Testing & Test Set Generation

A digital circuit responds to discrete potentials asserted at its input lines, and, as a result, asserts another set of these "digital signals" at its output lines. In general, the value of a digital signal is restricted (within a threshold) to logical 1, corresponding to power supply potential, and to logical 0, corresponding to ground. A single enumeration of input signals applied to a circuit is called an "input vector" or "input pattern" to that circuit, and likewise, the corresponding collection of output values is an "output vector" or "output pattern". A *combinational* circuit is one which does not contain memory, thus its output state depends only on the input vector applied. On the other hand, due to the presence of memory, successive states of a ***sequential*** circuit are related. This implies that a specific sequence of input vectors may be required in order to force a sequential circuit into a particular output state.

This chapter introduces some of the ideas involved in automatically generating vectors for testing combinational circuits. The same task, if performed with respect to sequential circuits is considerably more computationally intensive [Set85]. Fortunately though, through the use of existing "scan design" techniques (covered in chapter 3), sequential structures can be temporarily converted into combinational ones for testing purposes. Thus, sequential circuits in a scan-based design environment are also included in the discussion.

2.1 Failures and Fault Models

Testing is the process in which a circuit's function or structure is validated. This is done by attempting to force a circuit into a known state and comparing the observed result with the expected response. A mismatch of these values implies that the circuit contains a "failure" and is imperfect. Failures are physical anomalies within a fabricated circuit which cause it to malfunction. Notwithstanding design errors, they may be the result of imperfections in the fabrication process leading to device flaws including shorts between conductors, broken interconnects, improperly doped regions and missing contacts. They may also occur as in-service defects caused by, for instance, metal migration or power overload. Failures may manifest themselves as parametric errors or affect the steady-state of the circuit.

In order to generate tests, a "fault model" should be devised to be representative of many, if not all, failures which can occur. To this date, a single model which can characterize all possible defects has not been formulated. The industry standard for combinational testing is the "single stuck-at" model [Eld56] which assumes that under the influence of a fault, a line is expected to be held at a fixed logic value irrespective of the polarity of the driving signal. Thus the line is said to be 'stuck' at a logical 1 or 0. For example, a NAND gate with a stuck-at 1 (s-a-1) fault on its output will assert an output value of 1 regardless of its inputs.

A second stipulation of the single stuck-at model is that a faulty circuit is assumed to contain only 1 fault. If multiple faults were considered, in an n line circuit there could be up to $3^n - 1$ possible faulty circuit representations since a line can be s-a-1, s-a-0 or fault-free. When a single stuck fault is assumed, the number of possible faulty cases is reduced to $2n$.

Although the single stuck-at fault assumption has been sufficient in practice [Wil83], there are some CMOS defects which cannot be properly handled using this model. In particular, some faults introduce memory effects into the circuit, thus a specific ordering of test vectors may be required for detection. An example of this is stuck-open faults

[Wad78] and a class of bridging faults [Mei74] which create unwanted feedback paths within the circuit. Transition faults [She85] and crosspoint faults in PLA's [Smi79] are other faults which the stuck-at model does not explicitly take into account

Nevertheless, because of its computational simplicity and since practical experience has shown that a complete stuck-at test also tends to detect a large number of other fault types [Wil83] (termed "windfall" fault coverage), the fault model used for this research is the single stuck-at model.

2.2 Approaches To Testing – Functional & Structural Tests

There are two approaches to generating test vectors These are based on whether a functional or structural representation of the circuit is processed [Gra89] [Man89]

Functional testing usually requires in-depth knowledge of the circuit's operation but tends to neglect explicit hardware details. This approach hierarchically checks the circuit and internal circuit modules by verifying that their intended functionality and interaction as a unit is according to specification For example, functional testing determines if adders combine bits properly, if memories can be accessed and if ALUs perform all desired operations Functional tests are commonly developed for design verification

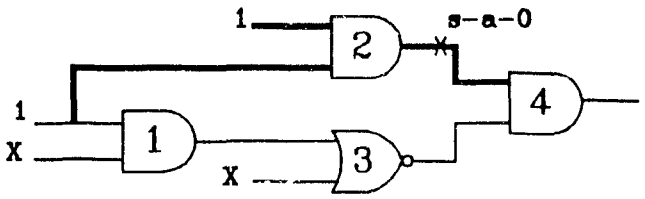
Structural testing attempts to give a level of assurance that the CUT will operate correctly by ensuring that its components (gates, lines, etc.) do not contain faults Development of these tests requires knowledge of the operation of only the basic circuit elements (e.g. logic gates and blocks such as flip flops and adders), and thus can be automated.

2.3 Test Pattern Generation

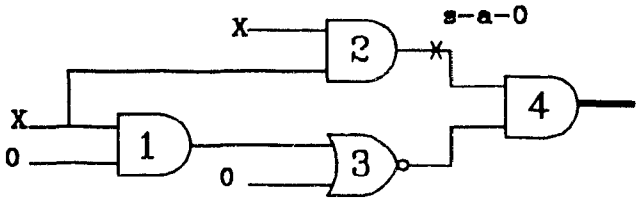
Testing a particular circuit node involves a path ***sensitization*** process in that an input test vector is generated which:

- stimulates the CUT inputs to force or *control* the desired node into the known fault-free state, **and**
- stimulates the CUT inputs so that the effects of this assignment can be propagated and *observed* at the circuit outputs

Figure 2 1 illustrates how parts of a test vector controls and observes a fault site In 2.1i, a s-a-0 on the output of gate 2 is to be tested. The input assignment 11xx controls the target line to a 1 (fault free value) Next, in order to observe the effects of the potential fault, a path must be sensitized from the faulted line to the circuit output Thus, the output of gate 3 must be a non-controlling value with respect to the output AND (gate 4) This is done in figure 2.1b by the input vector xx00 Since there is no conflict between the vectors required to control and observe the fault, the final test vector is 1100.



(i) – control



(ii) – observe

Figure 2.1 Operation of a Test Vector

Generating a test vector for a given fault belongs to a class of problems known as "NP complete". This means that the complexity of the task may increase exponentially with the size of the input (in this case the number of circuit lines) under consideration [Nil80]

As a result, heuristics (see references of section 2.3.1) are designed which attempt to reduce the amount on computational overhead encountered in developing test sets.

There are 2 approaches to generating a deterministic test set:

- Algorithmic test pattern generation
- Random pattern based test pattern generation

2.3.1 Algorithmic Test Pattern Generation (ATPG)

For each potentially detectable fault in a circuit, ATPG tools attempt to analytically design a test vector by implementing a path sensitization process. This idea has existed for roughly 2 decades now and originated with the D-algorithm [Rot66] Some improved methods developed through the years include [Sel68](boolean difference), [Goe81](PODEM), [Fuj83](FAN), [Ake76], [Sch88](SOCRATES) and most recently [Cox90](CAMP)

2.3.2 Random Pattern Based Test Pattern Generation

The premise here is that a sufficiently large number of pseudorandomly generated test vectors can detect most, if not all, faults. The typical result of this trial and error approach is shown graphically in figure 2.2.

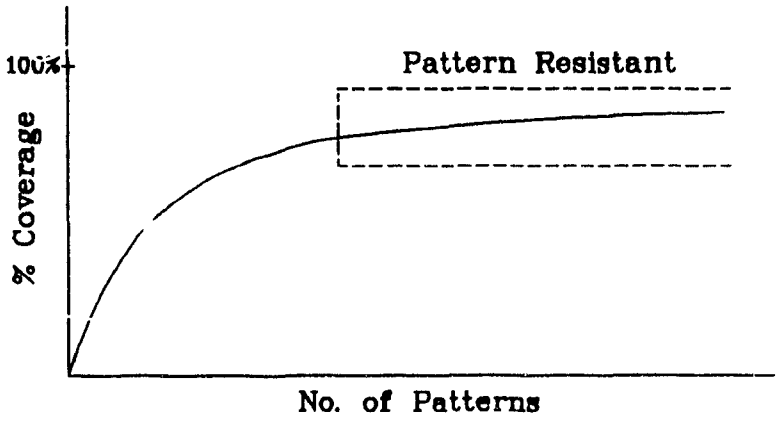


Figure 2.2 Progression of Coverage with Pseudorandom Patterns

It is common that a large portion of the detectable faults are covered within a relatively small number of input vectors, resulting in a large positive slope in the progression of coverage graph However, there exists a group of faults, classified as "pattern resistant" which, if detectable, may require test lengths orders of magnitude larger than the size of a deterministic test set The flattened tail region of the coverage graph corresponds to the relative test lengths after which pattern resistant faults are detected.

There are a few reasons why a particular fault might be pattern resistant [Tot88].

- Redundancy - the fault is not detectable
- Reconvergent fanout - there is a possibility of cancellation of fault effects
- High fan-in - introduced correlation tends to inhibit propagation
- Test vector quality - e.g the generated distribution of 1 and 0 assignments on the CUT inputs is in conflict with what is required to control/observe many faults, or there exists correlation between successive patterns or bits and this correlation prohibits/impedes fault detection

A potential remedy to the potentially unreasonable time penalty which may be incurred in detecting pattern resistant faults is a joint generation scheme - pseudorandom patterns are generated to detect a large group of faults after which ATPG is used to detect the rest [Kaw85]. As is demonstrated by this thesis, weighted random patterns can be also used to dramatically increase the rate of coverage compared to standard uniform random pattern generation.

One of the earliest applications of testing with a generated sequence is *fault injection* (also called *fault insertion* [Sus73]). This entails physically injecting a fault into a discrete model of the circuit built with 'off the shelf' components, followed by randomly[1] assigning a series of test stimuli. The test vectors which produce a response different than that of the known fault free circuit are retained as tests for the injected faults This classical method is suitable for small circuits (TTL, DTL) using low levels of integration but clearly such an approach is impractical with today's circuit densities. The method is mimicked through the use of computer simulation.

[1] in the course of this text 'random' or 'uniformly distributed random' generation refers to pseudorandom generation in which there is an equal chance of a bit being assigned 1 or 0

2.3.2.1 Simulation Based Random Test Pattern Generation (RTPG)

Similar to ATPG, a structural description of the circuit structure is constructed in software, and appropriate modeled faults are associated with each circuit node. For each pseudorandom vector applied, the simulator reproduces the result of the faulted and fault free circuit, and checks if any of the modeled faults are detected. If so, the detected fault may be removed from the original fault set, and the pattern is retained as a test vector.

Such simulation based approaches can be used to provide *compact* deterministic test sets. This is done in the following manner: The test vectors found nearing the end of the process detect very specific (more pattern resistant) faults and may also cover many of those 'easily detectable' faults found earlier in the procedure. By simulating the extracted test set in reverse order to which the vectors were found and pruning the fault set after each vector, the size of the test set can be reduced by about 25% to 50%. This procedure is commonly called *reverse compaction*.

The complexity of fault simulation tasks is believed to be $O(n^2)$ [Har87], where n is the number of circuit gates. This is mostly due to the existence of reconvergence which creates correlation between gate inputs. Some existing simulators are described in [Wai85] and [Maa87].

2.3.2.2 Non-Simulation Based Random Pattern Test

In order to eliminate the computational overhead of fault simulation, a 'sufficiently long' test sequence is applied to the CUT. Unlike section 2.3.2.1, although the test set is never formally designed, it is assumed that it is contained within the generated test sequence.

Since fault simulation is not performed, the fault coverage of the test sequence can only be approximated. Sometimes a statistical sampling method [Set85] is used to estimate this value. Here, a randomly selected sample of circuit faults is simulated using the sequence to be evaluated. The resulting fault coverage of this subset is used as an

approximation of the overall coverage.

An estimate of the required test length, N, can be found using the formula :

$$P_N(X) = \prod_{f \in F} (1 - (1 - p_f(X))^N) \quad (1)$$

where, $P_N(X)$ is the given threshold probability that each single fault f in the original fault set F is detected within a test sequence of N test vectors. $p_f(X)$ is the detection probability of the fault f [Gol74] [Brg84] [Jai84]. This is a measure of the odds of detecting the fault f, and is dependent on the relative distribution (X) of 1s to 0s at each bit position in the input sequence used. It has been found [Sav84] that only the faults whose detection probabilities are roughly within a factor of 2 of the lowest detection probability within the fault set affect the test length estimate.

At the begining of this chapter it was mentioned that sequential circuits can be temporarily converted into combinational structures. This idea of altering the structure of the CUT to ease testing tasks is elaborated upon in the following chapter.

Chapter 3 Design for Testability

It was seen in Chapter 2 that testing for a circuit fault depends on the ability to control and observe a fault site The test circuit used to demonstrate the idea (figure 2 1) was a fairly simple and small structure If however, the block was embedded in a much larger cell with a smaller pin to gate ratio, the ability to affect the fault site may be impaired As a result, testability analysis programs [Gol74] [Brg84] [Jai84] have been developed to provide approximate measures of the controllability and observability of each node within a circuit. While these measures are not sufficient to indicate whether or not a specific fault is detected, the information can be used to locate a section of the CUT which is potentially difficult to test [Agr82]. With this data, the circuit can be modified to enhance its overall testability. Also, since these algorithms are less computation intensive than those designed to generate deterministic test vectors, the data is available relatively quickly during the design process.

The concept of altering the circuit structure to simplify the test process essentially brings the test process directly into the design environment As such, an ever-growing knowledge base of design methods called, *Design for Testability* (DFT) [Wil83] is constantly being compiled. There are two DFT categories ad-hoc and structured. Examples of each of these design approaches are given next.

3.1 Ad-Hoc Approaches

Ad-hoc DFT methods are usually device-specific and not intended to solve a general

test problem. Two common strategies are logic partitioning and test point insertion.

3.1.1 Partitioning

Since the computation time for test pattern generation and evaluation is proportional to the number of gates squared for combinational circuits and cubed for sequential circuits [Set85], a divide and conquer approach is taken to help reduce the test generation effort. Essentially the entire structure is subdivided into separate circuit blocks during the test mode Depending on the interdependence between subcircuits, partitioned regions can be tested in parallel thus decreasing the overall test time

Modular or regularly designed cells with natural partitions are more appropriate for this strategy than are unstructured circuits designs. For instance, some designers exploit the natural inter-module isolation/communication of bus-based architectures to partition a circuit during test mode. However, with such an approach isolating bus failures tends to be a cumbersome process [How89]

3.1.2 Test Point Insertion

As mentioned previously, regions of the CUT which are difficult to access can be identified. The controllability and observability of these sections can be physically enhanced by inserting accessible *test points* which, during test mode, may act as circuit inputs, circuit outputs or both. A trivial example is to replace a circuit node by a flip-flop connected in a scan-chain (see sect 3.2). Selection of optimal insertion sites remains an open problem. Some results are discussed in [Hay74][Hay73][Iye89]

3.2 Structured Approaches

The intention of structured DFT is to introduce a design methodology to solve a general test problem. These techniques are usually geared for automated design.

Scan Design is probably the most popular structured DFT practice. It is based on providing access to circuit latches. There are several variations of the scheme, differing

in latch, clocking and control designs imposed by in-house system design rules [Wil73] [Ste77] [Eic78] [And80] [Nad88] [Fun89], but the underlying principle usually remains the same.

3.2.1 Full Scan Design

The intention of full scan design methodology is to test a complex sequential circuit by verifying that its structural build-up is sound[2]. In principle, for testing purposes, a sequential circuit is transformed into a combinational block, thus existing combinational test generation techniques can be employed. Such an approach is attractive because of the relative ease in generating tests for combinational as opposed to sequential circuits [Set85].

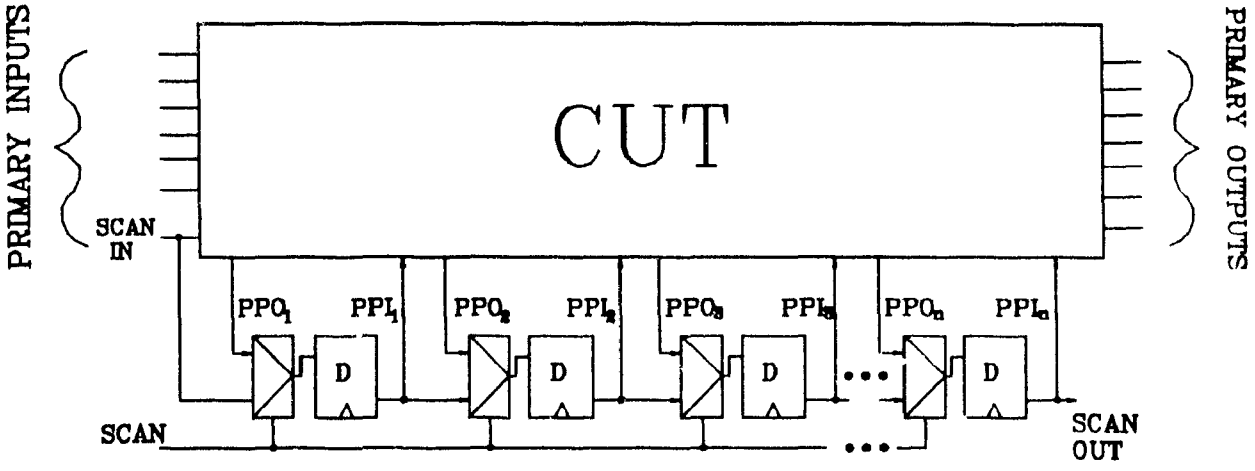


Figure 3.1 Full-Scan Design Concept

The basic full scan design concept is logically shown in figure 3.1[3]. Each system latch, shown here as a D type flip-flop (D-FF), is modified to accept multiplexed input one from the output of the previous scan cell, and the other from the circuit node previously associated with the original latch input. An extra control line (SCAN) is needed to globally regulate the input selection of the multiplexers. The result is the

[2] A limited amount of functional vectors are also done for design verification and timing analysis

[3] A good catalogue of techniques is contained in [McC85b] [Wil83] and [Fun89]

creation of accessible pseudo-primary input nodes (PPI_i - previously the latch output) and pseudo-primary output nodes (PPO_i - previously the latch input), and the provision of at least two modes of operation - **normal** and **scan**.

As the name implies, in **normal** mode the flip-flops (and thus the system) performs with the intended functionality and the test circuitry is virtually transparent to the CUT. In **scan** mode, each scan cell accepts input from the output of a predecessor scan cell thus collectively joining to form a large shift register structure or *scan-chain*.

During the test procedure, for each test vector, the system is first placed into **scan** mode and, aside from the bits which map to primary inputs, the input vector is serially shifted into the scan-chain. The CUT then returns to **normal** mode for 1 cycle so that the circuit's response to this excitation can be loaded into the scan-chain. Finally the system is again placed into **scan** mode and the test data contained in the chain is shifted out to be analysed, while another test vector is simultaneously shifted in.

The penalty for reducing the effort needed for test generation is an increase in area, higher power consumption and possible operating speed degradation if test hardware is included in a critical path. Also, since test vectors are serially shifted into the chain, the time needed to apply the test is proportional to the size of the scan-chain used. A partial solution is to introduce some parallelism by partitioning the scan-chain. Realistically, the number of these segments is limited by tradeoffs involving factors such as test time, area overhead and the pin limitations of both the external tester and chip [Bas89]

Some of the drawbacks of full scan design can be reduced by using a partial scan design [Pra88] [Che89] In this scheme, only a subset of the system latches are configured into a scan-chain. However, in such a system, the more difficult problems of sequential test generation and selection of optimal candidate scan cells must be considered

Despite the inherent disadvantages, many companies claim that the potential simplification of the test pattern generation process justifies the use of full scan design [Wil83].

3.2.2 Boundary Scan

A board–level extension of chip-level full scan design is boundary scan [Glo88] [Has88] [Par89]. In this technique the signals at the chip peripheries can be controlled and observed via scan cells associated with each of the primary input and output pins. Boundary scan promises many advantages, such as enhanced board-level diagnosis, more standardized tests, and reduced test access problems, especially for in-circuit testing.

As an alternative to the limitations imposed by the physical probes used in "bed-of-nails" testing, boundary scan offers more reliable isolation of circuit nodes and eliminates the overdriving of internal lines. The gain is dramatic with high density single or dual sided surface mount boards [Par89].

Another structured DFT technique is to include test pattern generation and test response evaluation circuitry to the chip or board under test. This is discussed in the next chapter. For further information on DFT see [Wil83].

Chapter 4 Practical Issues in Testing

As discussed in chapter 2, much work has been conducted on off-line generation and verification of test sets and sequences. The second part of the test process, as defined in chapter 1, is to devise a mechanism to physically apply the test vectors to the CUT and analyse the response

The goal is to be able to detect all modeled faults. However, because of possible redundancies within the CUT, the total number of detectable faults may not be known (identifying redundancies is an NP-complete problem) unless a large amount of preprocessing is done. So, the choice of a testing strategy may be based on a tradeoff of many inter-related variables including .

- a pre-defined target level of fault coverage
- production quotas
- time needed to perform the test
- on-chip area overhead introduced by DFT
- the effort required to implement DFT circuitry and perform the test
- test equipment and circuit maintenance costs

Of course, no test scheme is universally adaptable because of the unpredictable advances in design technology and fault modeling However, current market oriented dynamics should be accommodated. For example, with the current popularity of scan circuits, there will be a need for test strategies geared towards very dense circuits with thousands of serially accessed scan inputs/outputs [Bas89] Chapter 6 proposes a test generation/application procedure for such circuits.

As a precursor to this, in the rest of this chapter, some common methods used for test pattern application are outlined, followed by a short discussion of how the corresponding CUT response is analysed and evaluated. An introduction to on-chip testing is also presented.

4.1 Test Pattern Application

Test quality and test application effort have a notable impact on the cost of testing an integrated circuit. Most schemes can be categorized as:

- stored pattern testing, or
- hardware pattern generation

Ad-hoc *store and generate* and *programmed* schemes are sometimes added to the list but may be considered as extensions of the two approaches listed above.

4.1.1 Stored Pattern Testing

Stored pattern testing involves the application of specific deterministic test vectors, each of which provides an incremental level of coverage. Applying these vectors externally may require large storage capability and expensive test equipment, priced on the order of millions of dollars. The technique is straightforward and suitable for many current testing needs. However, the specifications of a test unit are static, so upper limits are physically imposed on such variables as test frequency, the number of available I/O channels, and the size of input (pin) buffers. On the other hand, device characteristics evolve in accordance with design and fabrication technology advances, and usually result in potentially higher operating speeds, higher pin counts and the need for progressively larger test sets. This conflict ultimately implies costly test equipment upgrades or replacement. It is questionable then, if long term testing is economically feasible with a standard stored pattern approach [Bas89].

4.1.2 Hardware Pattern Generation

In order to reduce the functional and memory requirements of the external tester required, relatively simple circuits, based on linear feedback shift registers (LFSRs) [Gol67] [Bar87] or cellular automata (CA) [Hor89], can be used to pseudorandomly generate test sequences. In the most straightforward case, vectors are generated such that there is an equal probability of assigning a 1 or 0 to a CUT input. As mentioned in section 2.3.2, it is intended that after a large number of these random vectors, a sufficiently high level of fault coverage can be achieved. Unfortunately, experience has shown (e.g. test lengths for C2670 and C7552 [ISC85]) that with current circuit densities, a number of these test vectors orders of magnitude in excess of the maximum test length permitted may be needed to attain this goal. Moreover a relatively small number of initial vectors cover a large portion of the detectable single stuck-at faults leaving the majority of the test sequence to be wasted in an attempt to detect random pattern resistant faults.

A possible alternative is to use a joint test strategy - apply a reasonable length of random test patterns and supplement this with stored pattern testing. However, it has been found that in many cases, the size of the stored test set needed nears 70% of the full deterministic set [Bas89] Thus, this approach does not address the problem of limited storage.

Another possibility is the use of a *store and generate* approach (e.g. [Agar81] [Abo83] [Bar85] [Fed86] [Eic87] [Ude88] [Brg89]) where, relative to the size of a deterministic test set, a small number control words are used in conjunction with a random pattern generator to produce a test sequence. *Programmed* testing is yet another extension whereby functional test programs contained in a ROM generate the required test set. This is common for microprocessor based chips and systems [Bra84] [Kuh83]

Exhaustive testing uses a counter, LFSR or CA to apply a full functional test set (i.e. 2^n distinct patterns where n is the number of circuit inputs). This approach is guaranteed to detect all combinational fault, but since the test length increases exponentially with the number of CUT inputs, this type of test procedure may not be economical for

circuits with with as few as 25 inputs. Thus partitioning can be employed to *pseudo exhaustively* [McC84] [Ude88] test the CUT as a set of smaller subcircuits. In such approaches, care is must be taken to minimise the sizable amount of control needed for in-circuit partitioning.

There also exists a small group of regularly structured circuits, whose functionality permits the ad-hoc design of very simple dedicated pattern generators. Parity tree testing [Hon81] is an example. However, such solutions are, by far, the exception rather than the rule.

As will be shown in Chapter 5, generating test patterns in which there is a non-uniform distribution of 1's to 0's can result in generated test lengths which are a small fraction of those otherwise needed for uniform random pattern generation. One drawback of most existing such weighted generation hardware is that while much simpler than stored pattern testing, these generators are much more complicated than their uniform random counterparts. However, chapter 6 describes a weighted generation scheme which balances hardware complexity and test length at the expense of an acceptable area penalty.

The root generator of many pattern generation schemes is a uniform random pattern generator often implemented using an LSFR or CA. The next two subsections introduce these structures.

4.1.3 Linear Feedback Shift Registers

A linear feedback shift register is a finite state machine comprised of a unidirectional chain of D-FF (unit delays) and XOR gates (modulo 2 adders). A typical LFSR with XOR elements positioned between selected memory stages, is shown in figure 4.1.

Such an LFSR generates a cyclic binary sequence by performing a linear transformation – polynomial division – on its input sequence (for more information see [Pet72][Bar87]). In implementing polynomial division, the feedback taps of an n bit LFSR define the

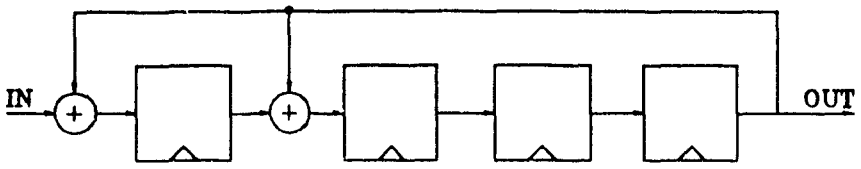


Figure 4.1 LFSR1 - polynomial divider - Characteristic Poly $x^4 + x^3 + 1$

divisor or "characteristic polynomial", and the input sequence (input polynomial) is the dividend. The n bit contents (or signature) of the unit after the last input bit has entered is the remainder polynomial and the sequence shifted out during the process is the quotient

A second LFSR structure is shown in figure 4.2. Here, a single XOR structure, positioned at the input of the first memory element of the LFSR, combines the previous states of selected LFSR stages.

Both figure 4.1 and figure 4.2 share the same characteristic polynomial. $(x^4 + x^3 + 1)$ and in terms of analysis, there exists a one to one mapping between the two LFSR forms. LFSR1 is a true polynomial divider, and while both it and LFSR2 yield the same quotient stream, their respective signatures may differ.

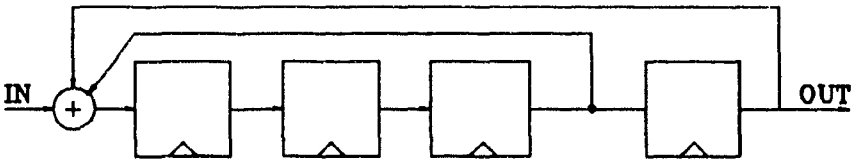


Figure 4.2 LFSR2 - Characteristic Poly $x^4 + x^3 + 1$

An LFSR is said to be autonomous if there is no external input stream. For an n-stage autonomous structure, the number of unique states cycled through is determined by the positions of the feedback taps.

If the characteristic polynomial is *primitive*[4], the number of unique internal states re-

[4] A partial list of primitive polynomials can be found in [Bar87]

peatedly cycled through is equal to $2^n - 1$ [5] In such a case the LFSR is said to be of maximal length.

A property of maximal length LFSRs is that the distribution of output bits generated is uniformly biased to 0.5 (the ratio of 1s to 0s occurring at each memory position over the maximal sequence is $\frac{2^{n-1}}{2^{n-1}-1}$). This property and their relatively uncomplicated structure, make LFSRs an attractive base for pseudorandom number generators

4.1.4 Cellular Automata

Another sequential structure which exhibits pseudorandom (uniformly distributed) generating traits is the cellular automaton. The unit is a series of flip-flops where the communication is restricted to nearest neighbours. The structure and communication of each block is defined by a set of linear "rules" [Wol83] which define the block's behavior. Two common examples are blocks constructed rule 90 and rule 150. (there are 256 possible rules). They are shown below where $s[t]$ is the value of position/state i at time interval t:

$$Rule\ 90: \quad s[t+1]_i = s[t]_{i-1} \oplus s[t]_{i+1} \tag{12}$$

$$Rule\ 150. \quad s[t+1]_i = s[t]_{i-1} \oplus s[t]_{i+1} \oplus s[t]_i \tag{13}$$

A hybrid CA is one which is constructed using more than rule.

[Glo88] [Hor88] [Ser88] have claimed that CA based generators possess superior randomness properties than LFSRs. In fact, [Ser88] establishes an isomorphism between the two structures which tends to ease analysis since much theory has already been developed for LFSRs.

4.2 Test Response Evaluation

There are many ways to analyze test responses. Conceptually, the simplest of these is to externally compare the circuit output with the corresponding fault free response found

[5] The all zero state is excluded since it forms its own cycle

through simulation. As described in [Dav76], CUT operation may also be compared to that of a fault-free reference unit.

If test response evaluation is to be done on-chip or on-board, or if it is desired to reduce, by orders of magnitude, the amount of data transfer between the CUT and the test head, data compaction techniques are employed. Compaction involves operating on the output data stream using some function which either performs a transformation or extracts some qualitative feature of the output data, or both. The most popular compaction scheme is signature analysis [Fro77] [Koe79] [Dav80] in which an LFSR or CA performs a polynomial division-like operation on the output stream. In this transformation, all possible input bit streams are mapped evenly onto the $2^n - 1$ possible signatures of an n-bit signature analyzer. It follows that the number of k-bit input streams which produce the same signature is :

$$\frac{2^k}{2^n} = 2^{k-n} \tag{15}$$

This raises an important point: since compaction is a reduction of information, there is an implied probability that useful knowledge is lost[6]. In the case of signature analysis using an n-bit signature analyzer, for every given fault-free signature, there are potentially $2^{k-n} - 1$ wrong bit sequences which could produce the same result. The phenomenon where an incorrect bit sequence yields the fault-free signature is called *aliasing*. It has been found that as the length of the input sequence tends to infinity, an n-bit LFSR or CA alias with probability 2^{-n} [Wil86] [Iva88].

Typically LFSR based signature analysers accept serial data. The extension for multiple output circuits is a multi-input signature register (MISR) which shown in figure 4.3 for a characteristic polynomial of $x^4 + x^3 + 1$.

Other compaction techniques involve recognizing certain attributes of the circuit's output data stream, such as parity [Car82], the ratio of 1's contained in the stream and its extension to exhaustive testing (syndrome testing)[Sav80], and the number of transitions occurring [Hay76]. Good overviews of various compaction techniques are given

[6] compaction should not be confused with 'compression' in which all information is recoverable

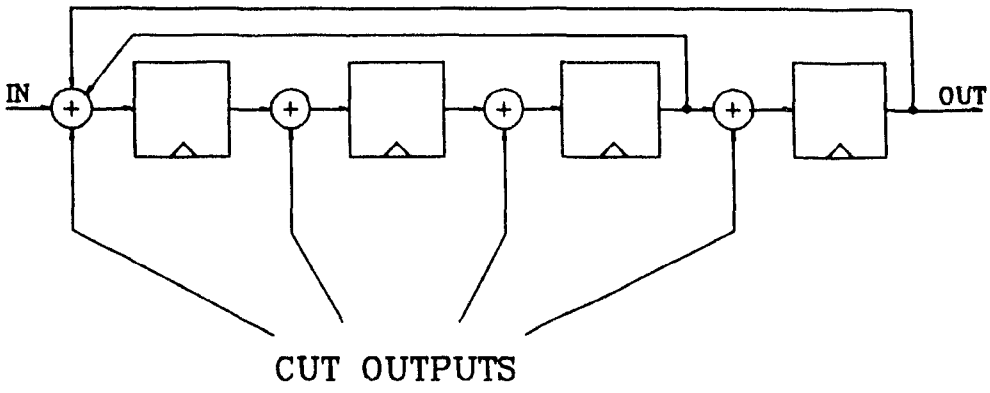


Figure 4.3 MISR - $x^4 + x^3 + 1$

in [Bar87] and [Kar85].

4.3 Built–In Self–Test

In Built-In Self-Test (BIST), test generation and response evaluation hardware are included on-chip so that in-circuit tests can be performed with minimal need of external test equipment, if any. The standard BIST set-up is shown in figure 4.4 Under normal operating conditions, the additional test circuitry is transparent to the functionality of the device.

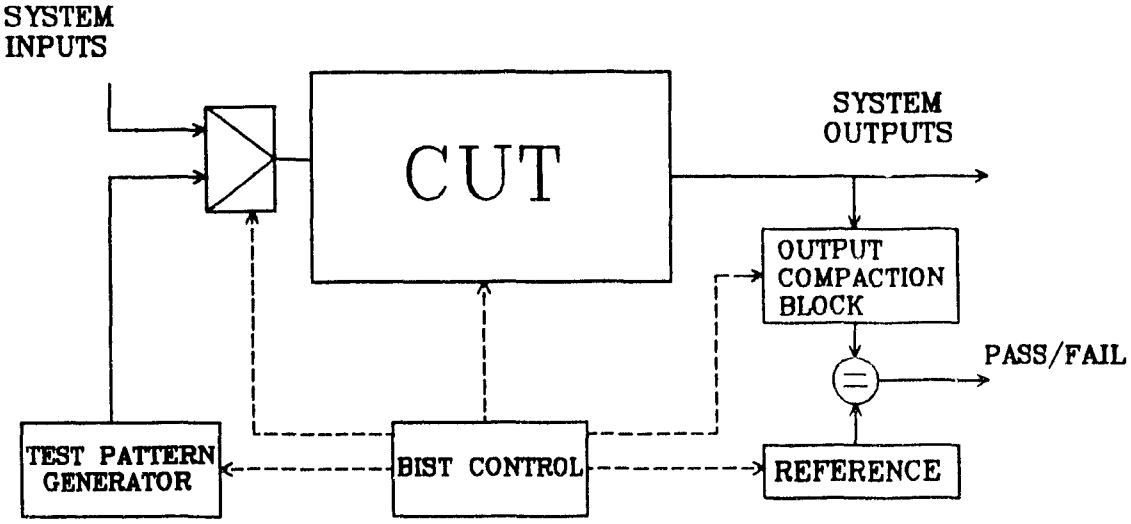


Figure 4.4 Standard BIST Scheme

During test mode, input stimuli are accepted from the test pattern generation block (TPG) and the CUT's output responses prepared for analysis in the output compaction block (OCB). The most popular compaction technique for BIST is signature analysis. At the end of the test session, the final signature(s) is/are compared with a reference value(s) and a result flag is signaled. This type of process typically gives a global pass/fail diagnosis (if desired [Ait89] suggests inclusion of more refined fault location features).

The reason for pursuing a self-test option is ultimately to reduce test costs, and there are, indeed, many potential advantages offered by such a strategy. For example :

- Less costly external test equipment is needed, thus the associated capital cost is reduced.
- Placing the application and verification circuitry within the CUT eases the test access problem.
- Less dependency on physical leads increases the efficiency of testing at other levels of packaging (e.g. probe tests and board tests) [Bar87] [Par89]
- Shorter test times are possible since some tests can be run at circuit speeds [Kra87].
- Potentially lower (test) run-time costs because of possible circuit partitioning
- Increased portability of testing (especially in-field test) allowing, lower long term maintenance costs (e.g user-initiated tests for microprocessors [Kub83])

However, some of the obvious hurdles which thwart the acceptance of BIST are :

- The extra area needed to implement the test circuitry reduces device yield.
- There is a potential performance degradation if test circuits are included in a critical delay path.
- As with external pseudorandom testing, excessive generated test lengths imply large time penalties.
- Common fault diagnosis techniques may be unattractive because, for example, constructing fault dictionaries for circuits with compaction is expensive [Ait89].
- There is a general lack of knowledge and design techniques to solve the above problems.

Proponents of BIST claim that the long term benefits offered by this test option far outweigh the drawbacks - in some cases even if the area overhead is 40% [Tot88] Furthermore self-test methodologies may be the only solution to test access problems which arise from ever shrinking chip and board-level packaging technologies [Par89] For a good introduction to BIST see [McC85a] [McC85b] [Bar87]

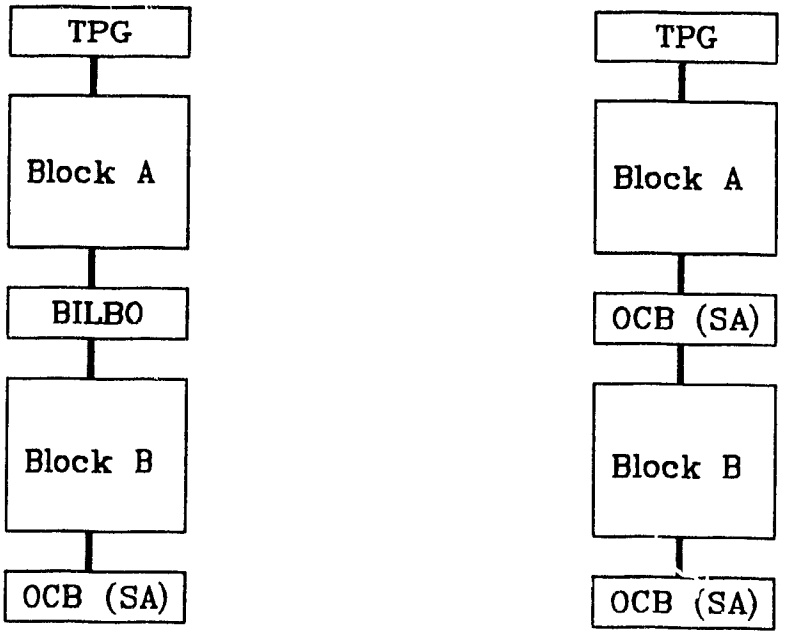
Since this thesis presents an on-chip test pattern generation technique, the next section introduces two pattern generators used for BIST.

4.3.1 Some BIST Test Pattern Generation Methods

The TPG is usually implemented as a variation of simple random pattern generating structures, such as LFSRs or CAs, configured to generate a maximum length sequence Sometimes, for example in the microcoded BIST of microprocessors, memory is included into the TPG Whether or not memory is included depends, however, on the amount of implementation overhead which can be tolerated.

One of the milestone structures proposed for BIST is a multifunction shift register called a "built-in logic block observer" (BILBO) [Kon79] The BILBO, and its cellular automaton counterpart, CALBO (cellular automaton logic block observer [Hur88]), facilitate partitioning the CUT into separate blocks for testing In a particular test session, the unit acts as either a TPG or a signature analyzer (SA). Figure 4.5a is a typical BIST set-up using a BILBO. Circuit blocks A and B are tested in separate test sessions in the first, the BILBO acts as a compacter for circuit A, and in the second it acts as a pseudorandom TPG for B. In this case, since separate blocks share the BILBO for different purposes, the blocks A and B cannot be tested in parallel

In order to reduce the test time and hardware overhead incurred when implementing a BILBO, some authors (e.g. [Kra87][Kim88]) suggest the use of signature registers as TPGs. Such a system can increase parallelism when testing separate blocks (provided there is no feedback between the blocks) This is demonstrated in Figure 4.5b With this setup, circuit blocks A and B can be tested in parallel because each intermediate



(a- Circuit Partitioning using BILBO) (b- Parallel testing of A and B)

Figure 4.5 BIST of Separate Circuit Blocks

signature provided by the OCB of A are used as test patterns for block B.

Circular BIST ([Kra87]) extends the idea of using signature registers for test pattern generation by creating a feedback shift register, where the CUT is the feedback logic, to simultaneously generate and compact test patterns for the CUT itself. Some attributes of this scheme are: tests can be run at system speeds, the entire circuit can be tested in one test session, and the overhead may be less than that for a standard BILBO scheme. In order to reduce the area penalty of implementing a full scan circular path, [Pra88] documents a method for selecting appropriate flip-flops for a partial scan circular self-test path. The effects of phenomena such as fault masking should be evaluated in a circular BIST scheme.

Chapter 5 Testing With Weighted Random Patterns

Conventional uniform random pattern generation, attempts to cover all faults within the CUT by generating a sequence in which there is an equal chance of each CUT input's assignment being 1 or 0. Experimental results [Sch75] [Sav84] [Lis87] [Wai88] [Wun88] [Bas89], however, suggest that generating groups of test vectors such that there is a non-uniform distribution of 1's to 0's on the input lines, can result in test lengths orders of magnitude shorter than that incurred when using uniform random pattern generation alone. In such a generation strategy, the proportion of times an input line assumes a value of 1 is called the *weight* or *bias* of that line, and the complete distribution of biases for a group of CUT inputs is called a *weight set* or *weight distribution*. Thus, according to the above definitions, a uniform random pattern generator possesses a characteristic weight set in which all pin biases are approximately equal to 0.5. Test pattern generation according to one or more non-uniform weight sets is called weighted random pattern (WRP) generation.

This chapter is divided into two sections : First, some of the known algorithms used to determine circuit specific weight sets are described; followed by a discussion of some of the current schemes used to implement WRP generation in hardware.

5.1 Some Algorithms to Determine Weight Sets

Unlike the new method proposed in this thesis, some documented methods used to

calculate circuit specific weight sets involve one or more of the following techniques to gather information about the circuit function or structure: correlation of internal switching activity to specific pin activity, formal detection probability calculations, and circuit path tracing. The salient features of some of these methods are discussed below.

In some earlier work intended for large-scale integration (LSI) proposed by Schnurmann et al. [Sch75], the relative weight of a particular input pin is adaptively assigned based on the relative amount of internal switching activity caused by uniquely exercising that pin. With respect to VLSI circuits, Siavoshi [Sia88] proposes an alternate adaptive algorithm which reduces the prohibitive amount of simulation needed in [Sch75] to evaluate nodal activity. Instead of randomly toggling single input pins, most of the initial information is gathered using a functional test set which already exists for design verification purposes. By simulating each pattern of this set, the amount of activity per vector is assessed from observing the circuit inputs in conjunction with the corresponding output states. This information is then used to develop an initial weight set from which weighted test patterns are generated and simulated. If any new test vectors are found as a result of WRP simulation, the estimated weight set is iteratively refined using a similar process. A potential drawback of the method as presented is that a functional test set may not always be a proper choice for the prime source of information, especially if it provides poor fault coverage. In such cases, the initial *information* may be biased resulting in the final test sequence being geared to test only a portion of the circuit.

Recently, Wunderlich [Wun88] developed a detection probability based procedure to determine a user specified number, k, of weight sets. This is done by modifying the common test length formula (equation 1) to the form shown below (equation 2), which accommodates for k generator weight distributions $X_1 \ldots X_k$ and k fault sets, $F_1 \ldots F_k$.

$$P(X) \leq \prod_{i=1}^{k} \prod_{f \in F_i} (1 - (1 - p_f(X_i))^{N_i}) \qquad (2)$$

where

$$N = \sum_{i=1}^{k} N_i \tag{3}$$

If only P is known, minimising the overall test length N is an NP-hard task [Kri84] So, k is made a free-variable and the new objective is to find a sufficiently small test set. The problem is formulated as follows:

> Given a probability P that all faults are detected, let k be the desired number of generator distributions Find k partitions of the fault set $F = < F_1 \quad F_k \quad >$ k associated weight distributions $X = < X_1 \quad X_k >$ and k test lengths $< N_1 \quad N_k >$ such that the total test length N is sufficiently smaller than that of a uniform random test

The procedure for determining multiple weight sets is an extension of the work contained in [Wun87] which solves the case of a single weight set.

Single Weight Assumption

For a single fault set (i.e. k=1), an optimal solution for X can be found by minimising the objective function (OF) -

$$ln(P(X)) \approx \sum_{f \in F} -(1 - p_f(X))^N \approx - \sum_{f \in F} e^{-p_f(X)N} \tag{4}$$

$$OF = \sum_{f \in F} e^{-p_f(X)N} \tag{5}$$

but this task generally requires exponential effort.

However, since the OF can be proven to be strictly convex with respect to a single variable[7], heuristically, a suitable distribution X can be found using Newton iteration (or similar method e.g regula falsi) to minimise the OF for each pin bias x_i.

Multiple Weights

The procedure for developing multiple weight sets involves partitioning the fault set F into k subsets $< F_1 ... F_k >$ of possibly differing sizes, such that the sum of the k

[7] Note that $P(X)$ is really a function of m variables since the weight X is a set of input biases $< x_1 \quad x_m >$ where m = no CUT inputs

associated OFs is sufficiently small. A weight is calculated for each separate fault grouping.

Again, the problem of finding an optimal division of the fault set requires exponential effort. Instead, a multi-stage partitioning scheme is adopted. First, a subset of faults with the smallest detection probabilities (see [Sav84]) is coarsly partitioned into k groups using a first order algorithm afterwhich the contents of these k fault groups then refined using a search tree. Each remaining fault is then assigned to one of these subsets. A complete description of the method is contained in [Wun88].

	ESTIMATED LENGTHS		SIMULATED LENGTHS				
NETLIST	Random [Wun88]	Weighted [Wun88]	Weighted [Wun88] Test len.	# Sets	Random (Tulip)		
C432	1.9e3	1 1e3	494	2	1088	608	1024
C499	1.2e3	1.2e3	1381	1	960	1216	1600
C880	2.4e4	710	578	3	21568	6848	9760
C1355	2 1e6	2 1e6	5288	1	3104	3744	1824
C1908	5.1e4	2.1e4	1074	6	10592	6688	15072
C2670	8 8e6	8.1e5	47110	5	3.7e6	4 3e6	6 1e6
C3540	8.0e5	1.1e5	11233	5	49248	84416	9280
C5315	4.0e4	1.1e4	4430	5	1920	2656	4128
C6288	6.3e2	2.4e2	239	1	128	160	192
C7552	3.1e11	2.8e5	24037	6	> 1e6	1e6	1e6

Table 5.1 Comparison of [Wun88] Data to Random Pattern Simulation Results

Based on the results contained in [Wun88], the performance of this method is difficult to evaluate. Some of the data is reproduced in table 5 1[8] Here, the estimated uniform random test length and the estimated 'optimal' weighted test length are shown against the number of simulated weighted patterns[Wun88] and three uniform random test lengths found via simulation (Tulip [Maa88]).

[8] For C7552, after weighted simulation 27 faults were left undetected

The following observations are made :

- ⊙ Compared to the simulated random test lengths, the equiprobable test length estimates seem excessive. In fact, in all but two cases (C2670 and C7552) the gain of using the determined weight sets is not significant This is possibly because the estimates attempt to give a confidence level that some large percentage (eg 99%) of all test sequences of this length will achieve 100% coverage
- ⊙ The weighted test length estimates for 4(C1908, C1355, C2670 and C7552) cases differ from the weighted simulation value by at least an order of magnitude
- ⊙ The major advantage of using WRP is to diminish the test length needed to achieve a high level of fault coverage Aside from C2670 and C7552, all of the ISCAS85 [ISC85] test circuits are fully testable in a relatively short pseudorandom test length thus don't benefit from the scheme The weighting algorithm of [Wun88] could be better evaluated if experiments were performed on circuits which require much longer random test lengths

Also, contrary to what is logically expected, it is found that when using this method, the test length does NOT monotonically decrease with an increasing number of determined weight distributions This in itself is a serious failure on the part of the algorithm The author attributes this to the inherent complexity of the problem and the approximations used in the applied heuristics The method's sensitivity to detection probability accuracy and the size and contents of the initial fault list may be another source of error. For example, if the contribution of redundant faults is not neglected, the calculated detection probabilities of this class of fault would corrupt intermediate test length estimates and result in inaccurate weight sets and a pessimistic overall test length projection

Bardell, Savir and McAnney [Bar87] discuss a path tracing algorithm to find an initial weight set. This is done based on the signal probability assignment to the inputs of a gate, and calculated according to the formulae given below Here, p_i is the bias of the inputs to the gate, p_o is the output signal probability, and k is the number of gate inputs.

The weight of a fanout stem is calculated as the mean of the signal probability assign-

BLOCK	p_i
AND	$p_o^{1/k}$
OR	$1-(1-p_o)^{1/k}$
INV.	$1-p_o$
NAND	$(1-p_o)^{1/k}$
NOR	$1-p_o^{1/k}$

Table 5.2 Backtrace Signal Probability Update Formulae [Bar87]

ments at each of its branches This, however, may not always be the most appropriate compromise because the relative "importance" of each branch is not considered. For instance, due to the averaging process, the contribution of a branch bias intended to cover a relatively large number of circuit faults, may be outweighed or cancelled (the average reverts to 0.5) by the bias(es) of one or more other branches, whose cumulative number of associated faults is far less than that of the previous branch A solution might be to rank each branch according to the size of the sub-circuit, thus the number of circuit faults, with which it is associated. The bias of a stem can than be estimated as a weighted average of the biases at each of its branches.

The weighting procedure of [Bar87] is as follows :

For each gate in the circuit, an initial weight is assigned to each of its n inputs according to the formulae·

$$AND, NAND : p_i = (n-1)/n \tag{7}$$

$$OR, NOR : p_i = 1/n \tag{8}$$

and the gate's contribution to the signal probability assignments at the CUT inputs is computed by propagating back the gate's output signal assignment along all possible backtrace paths, and updating according to the equations outlined in table 5.2.

As a result, after all gates are processed, each CUT input will have m assigned local weights, where m is the number of possible gates which can be traced to that CUT input. The bias of a CUT input is then calculated as the average of all assigned local

weights. See [Bar87] for an example of this.

Multiple weight sets are found in the following manner: Using the initial weight set calculated above, a threshold number of weighted test vectors are simulated. The faults left undetected after sequence are identified. ATPG is performed to determine a test vector for one of these remaining faults and a test weight is computed according to the form of this new vector. That is, for each logical 1 in the test, a sufficiently high signal probability (e.g. 0.9) is assigned to that input position. Similarly, input positions corresponding to 0 values are assigned a low weight (e.g. 0.1) So, the test vector 1xx001 may produce the weight (0.9, 0.5, 0.5, 0.1, 0.1, 0.9). Again, simulation is performed and the fault list is further reduced. The ATPG-simulation based procedure repeats to find other deterministic weights[9].

An intermediate procedure which can be used in conjunction with the ATPG step to find multiple weights is to iteratively perform the initial signal probability based weighting algorithm on progressively reduced fault sets. Thus, dedicated weights will be algorithmically found for faulty 'sections' of the CUT This would be beneficial if weights defined by different subcircuits conflict, and tend to 'smooth' parts of the final weight set back to a near uniform random distribution.

An algorithm which uses such an idea is proposed by [Lis87]. Test generation system ESPRIT (Enhanced Statistical PRoduction of Test vectors) is designed to use testability measures and feedback form fault simulation to dynamically generate test vectors according to multiple weights

In this approach, testability estimates [Brg84] are used to calculate input probabilities by minimising a cost function which balances CPU time and simulated test length Patterns are generated according to the assigned weight set and fault simulation is done until a threshold rate of fault coverage is reached. Using the resulting reduced

---

[9] In such methods, it is always an option to store a few test patterns rather than implement a pure weighted random test strategy

undetected fault list, (thus for simulation purposes, a reduced representation of the circuit) the process repeats to find a supplemental weight distribution This procedure of determining weights and updating the fault list continues until a threshold level of fault coverage is attained

Experimentally, compared to a random pattern test length, the scheme is found to successfully diminish the generated test length needed to test the ISCAS85 benchmark circuits[ISC85]. The results of [Lis87] did not, however, discuss the number of weight distributions computed during the procedure

In an algorithm patented by IBM [Eic87], all possible paths from an output terminal to an CUT input are traced and an initial weight is calculated as a function of the number of CUT inputs feeding the blocks along the trace-back paths. Weighted simulation is then performed and DTPG is used to find test vectors for any faults left undetected Additional weight sets are derived from the DTPG results. The procedure for estimating the initial weight is discussed below :

In this algorithm, **each** input of a circuit block ,X, shares the **same** weight value The individual device blocks are limited to AND, NAND, OR and NOR primitives. Assuming that identical circuitry feeds each of a block's N inputs, the probability, P, of placing a non-controlling value on any block input is approximated by (no justification for this was given in [Eic87]):

$$P_{min} = (3 - N - (N^2 - 2N + 5)^{1/2})/(2 - 2N) \tag{9}$$

It follows that the ratio of occurrence of a non-controlling setting to controlling assignment is

$$R_{min} = P_{min}/(1 - P_{min}) \tag{10}$$

For example, in order to minimise the generated test length needed to test all stuck-at faults on a 4 input OR, the probability of placing a 0 on each gate input should be 0.768. In accordance with the definition of weighted generation given previously, the bias of each input line should then be 0.232.

Since identical input circuitry is a rare occurrence, a compensation factor is introduced based on the number of CUT inputs which can be traced to the block X. This is calculated as :

$$K = 0.5(NDI_x/NDI_y) + R_{min}(N_x) \tag{11}$$

where,

NDI_x = No. of CUT inputs controlling X

NDI_y = No. of CUT inputs controlling the block Y which feeds X along the selected traceback path

N_x = No. inputs to X

Equation (11) can be interpreted as the ratio of non-controlling to controlling input probability being offset by the ratio of the number of CUT inputs feeding the source block X, versus the number of CUT inputs feeding the block (Y) which is input to X.

Each circuit block is assigned two numbers $W1$ and $W0$, which represent the 1 and 0 weights shared by all of the block's input lines. These values are initialized to 1, and due to the different functionality of the 4 block types, they are conditionally updated as in table 5.3.

BLOCK	$W0$	$W1$
AND	$W0_x$	$K/W1_x$
NAND	$W1_x$	$K/W0_x$
OR	$K * W0_x$	$W1_x$
NOR	$K * W1_x$	$W0_x$

Table 5.3 Weight Number($W1$,$W0$) Update Calculation [Eic87]

Starting from each CUT output, a recursive backtrace is performed, progressively updating the $W1$s and $W0$s of the input blocks encountered along the backtrace paths. The $W1(W0)$ of an input block, Y, is modified as the larger of the $W1(W0)$ value calculated from table 5.3, and the $W1(W0)$ already assigned to the input block. For example if block X has a NAND as an input circuit block, the Ws for the NAND (i.e. block

Y)are updated as :

$$W0'_y = MAX(W1_x, W0y) \tag{12}$$

$$W1'_y = MAX(K/W0_x, W1_y) \tag{13}$$

The backtrace then continues from the NAND. When an CUT input is reached, its bias is indicated by the ratio of the $W1$ and $W0$ assigned to the line. For example if $W0$ $W1$ then the line is more strongly biased towards 0. The reader is referred to [Eic87] for further details.

Note that the compensation (equation. 11) is not accurate for circuits with fanout and/or reconvergence. The root of the problem is the assignment of identical $W1$ and $W0$ values to each of a block's input lines.

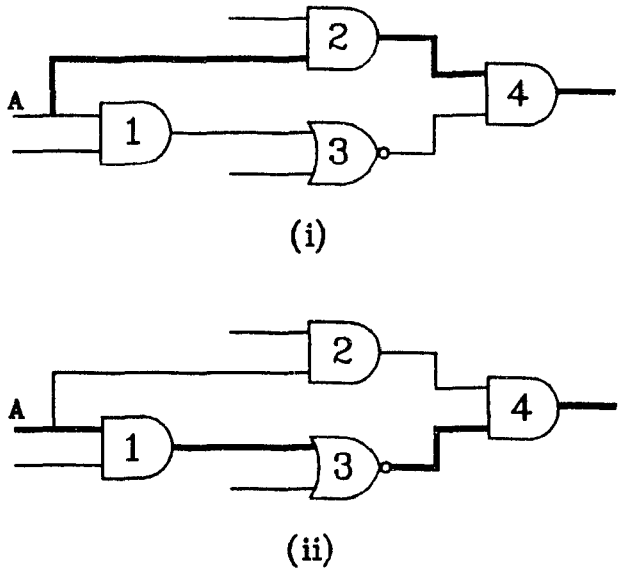


Figure 5.1 Failure of [Eic87] due to Fanout

If a particular circuit line fans-out, then it is shared by more than one subcircuit and hence, more than one backtrace path. It follows that the weight assignment of the fanout stem dictated by each backtrace path may be different. Figure 5.1 is an illustration of

this. The fanout stem A is shared by gates 1 and 2, thus the calculated Ws for stem A may be conflicting depending on which of the backtrace paths shown is taken.

Regardless of the inherent inaccuracies of the respective weight estimation processes, the acceptable performance of weighted test pattern generation suggests that the weight sets do not have to be optimal. In a practical sense, since thousands of test patterns are generated, the weight must simply represent enough information so that its performance is sufficiently superior to uniform random pattern testing

5.2 Weighted Generators

This section outlines some of the known implementation schemes used to realize WRP generation in hardware. Depending on how input stimuli are mapped onto circuit inputs, hardware WRP generators can be divided into two classes of architecture:

- Local generators, and
- Global generators

5.2.1 Local Generators

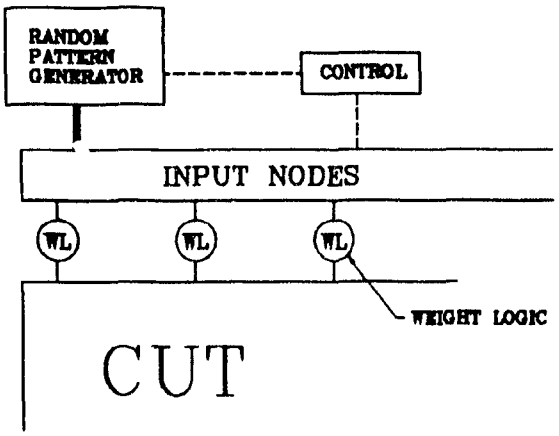


Figure 5.2 Local Weighted Pattern Generation

In local generation, weighted inputs are computed and applied directly at an input node

by logic associated with that node (figure 5.2). Since weight conversions are performed at the input sites, the hardware overhead for this type of architecture tends to grow as a function of the number of weighted CUT inputs required. This implies that the weight logic assigned to each CUT input should be simple. Moreover, the size and complexity of hardware required to locally generate weighted patterns according to multiple non-uniform weight sets may be unacceptable for implementation

Some of the earliest work in external local weighted pattern generation is contained in [Sch75] and reproduced in figure 5.3. Here, the weight at a CUT input reflects the relative amount of switching it undergoes. This however, does not detract from the previous definition of weighted generation given at the begining of this chapter

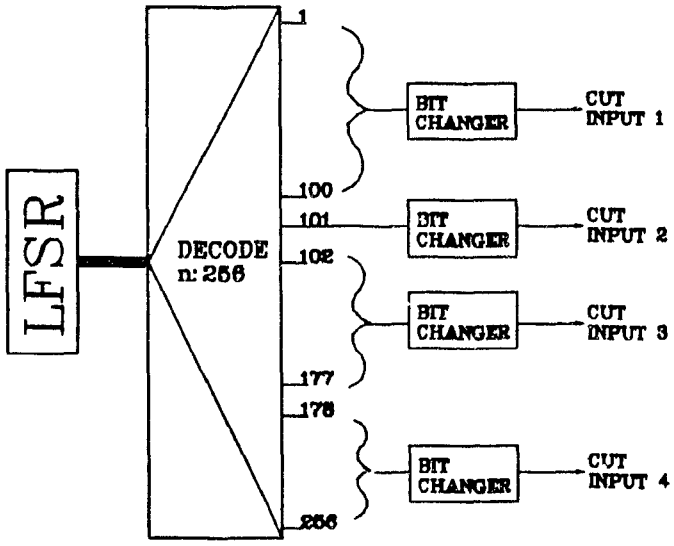


Figure 5.3 Weighted Generator of [Sch75]

An LFSR is used to stimulate each input of a large decoder a uniform number of times Each generator output is driven by a "bit changer" or binary trigger, which toggles each time a signal is applied to it Thus, a relative weight is assigned by driving a CUT input's bit trigger by a bundle of (ORed) decoder outputs. For example in figure 5 3, CUT $input_1$ is weighted to switch 100 times more than CUT $input_2$ and about 1.3 times more than CUT $input_3$ and CUT $input_4$.

This structure is actually a good example of how technology changes impact test hardware. The technique may have been appropriate for circuits of the mid-70's (the test circuits in [Sch75] contained less than 50 inputs) but nowadays, the comparatively large number of circuit inputs resulting from techniques such as scan design, and the associated serial manner in which most of these CUT inputs are accessed, renders this generation scheme infeasible. Incorporating the generator on-chip is impractical due to the high degree of routing necessary.

5.2.2 Global Weighted Pattern Generators

The main characteristic of global WRP generators (figure 5.4) is that weighted bit streams are mapped onto the appropriate CUT inputs. Frequently, the actual pattern generation circuit is autonomous to the CUT and a control block is needed to perform the mapping. An advantage of this architecture is that the control block can be constructed to enable test pattern generation according to multiple weight sets. In such cases, however, the size of this block may make the global WRP generator unattractive for full on-chip implementation.

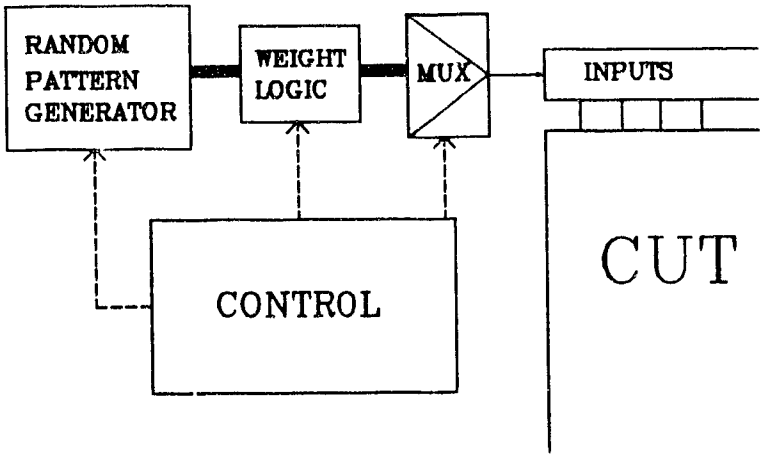


Figure 5.4 Global Weighted Pattern Generation

[Fed86] uses a memory based global WRP generator for the testing of microprocessors

First, a feedback shift register is designed such that there is no correlation between subsequent states. As shown in figure 5.5, the uniform output of this n-bit FSR is decoded to non-uniformly address a store (ROM) of k instruction words, where 2^n is greater than k Thus, complete instruction patterns (not bits) are weighted by mapping a desired number of FSR states to that pattern's address in the ROM. The method is well suited to off-chip functional testing but the ROM is may be too large for inclusion on-chip.

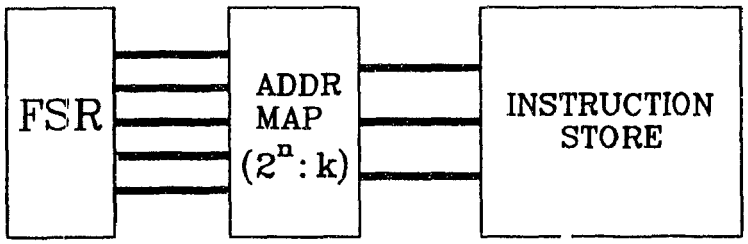


Figure 5.5 Weighted Instruction Generator

[Wun87] suggests a generator of unequiprobable random tests or GURT. This multi-function register unit has 4 modes of operation -

- normal system operation wherein each register serves as a D-flip-flop.
- a shift register
- a signature analyser
- a weighted generator

Of interest to this text is the last mode.

As a weighted generator, each GURT is configured to generate a 1 according to three probabilities p, 1-p and 0.5. This is done in a manner similar to combining the random bits contained in distinct LFSR stages through a small combinational block, the output of which is biased to the desired weight In some cases, in order to implement different weights (eg. 1/4 and 1/16) multiple GURTs are constructed A major drawback of this test architecture lies in the mapping of the weighted stream(s) onto circuit inputs.

Since a GURT can only generate 3 weights, input (scan) nodes of similar bias must be grouped to minimise routing costs and avoid repeatedly constructing GURTs of the same weight. However, weight sets are usually developed after the CUT is designed and

the resulting assignment of bias levels to CUT inputs can be in any order. Post-design reordering of scan cells to regularize the weight assignments is not a simple task and can (and probably will) introduce a severe routing overhead. Replicating CURTs of the same weight may also introduce an unreasonable area penalty

[Eic87] describes a weight generation system which is integrated into an external tester The basic weighting circuit is shown in figure 5.6 WRP generation is based on combining (ANDing) the random bits output from up to 5 consecutive stages of a 32-bit LFSR.

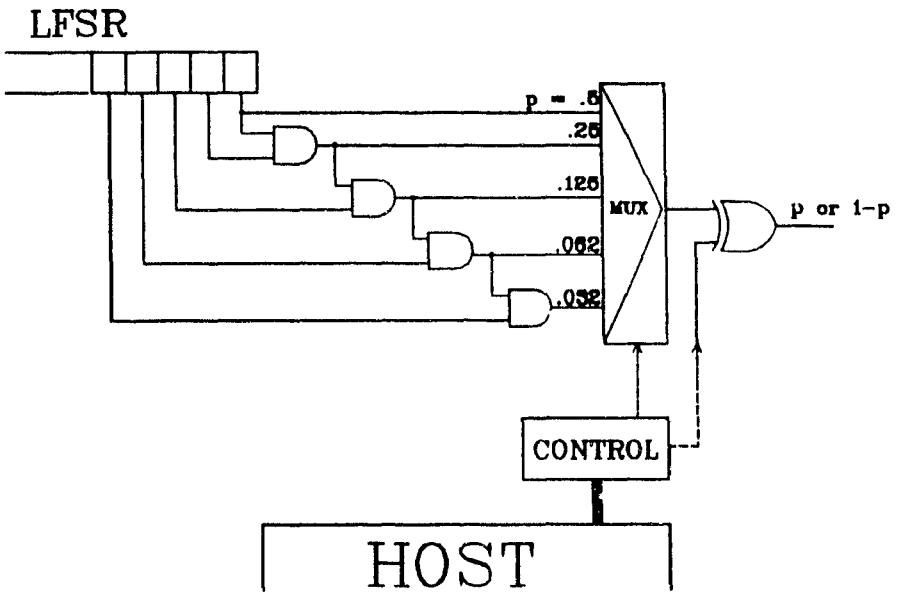


Figure 5.6 Cascade AND Weight System

Stream biases of 5 bit resolution are possible. The cascaded AND structure computes biases of 2^{-i}, i=1...5. The output XOR is used to logically invert the generated stream thus provide complementary biases of $1 - 2^{-i}$ for the same range of i

In order to reduce interdependencies within the generated bit stream, a delay-shift method of generation is used to stagger the spatial rate at which bits are combined For instance, if a bias of 0 5 is required, line #1 of the output mux would be selected every shift cycle. On the other hand, because a bias of 0.125 is the product of ANDing

3 consecutive LFSR stages, when needed, line #3 is gated through the MUX only after every 3rd cycle. The other non-equiprobable biases are generated in a similar manner.

Each output channel of the external test unit has a dedicated weight circuit, the LFSR of which has unique feedback taps and a unique initial seed This is done to help ensure that a pair of independent device inputs do not repeatedly receive identical test stimuli The sizable amount of control information needed to select and map biases onto input pins is handled by an external host computer.

Another global pattern generation system based on a "canonical weighting circuit" is suggested by [Brg89] and shown in figure 5 7 The weighting circuit is a cascade of r multiplexers whose select lines are determined by r distinct taps of a maximal length pseudorandom source biased to 0.5. Dedicated to each of scan input (S_j)of the CUT is a control word ($W^r W^0 ... W^{r-1}$) which is input to the multiplexer bock and designates the weight of the scan cell according to :

$$Weight(S_j) = 2^{-r}W^r + \sum_{i=1}^{r} 2^{-i}W^{r-i} \qquad (14)$$

In this way the unit generates patterns with a weight resolution accuracy of 2^{-r}. The reader is referred to [Brg89] for a detailed discussion of the weight derivation.

The size memory control block may tend to prevent the entire system from being integrated on-chip, however, on-board implementation is possible. The memory is expected to grow as $O(knr)$, where :

- k = number of weight sets
- n = number of scan cells.

Thus, chips with the order of 1000 scan inputs and 2 weight sets of precision 2^{-4}, require the order of 10K bits of storage. Since test time can be traded for hardware complexity, a more coarse weight set may be chosen in order to reduce the size of the memory.

The design of a new on-chip local WRP generation mechanism is presented and evaluated in the next chapter.

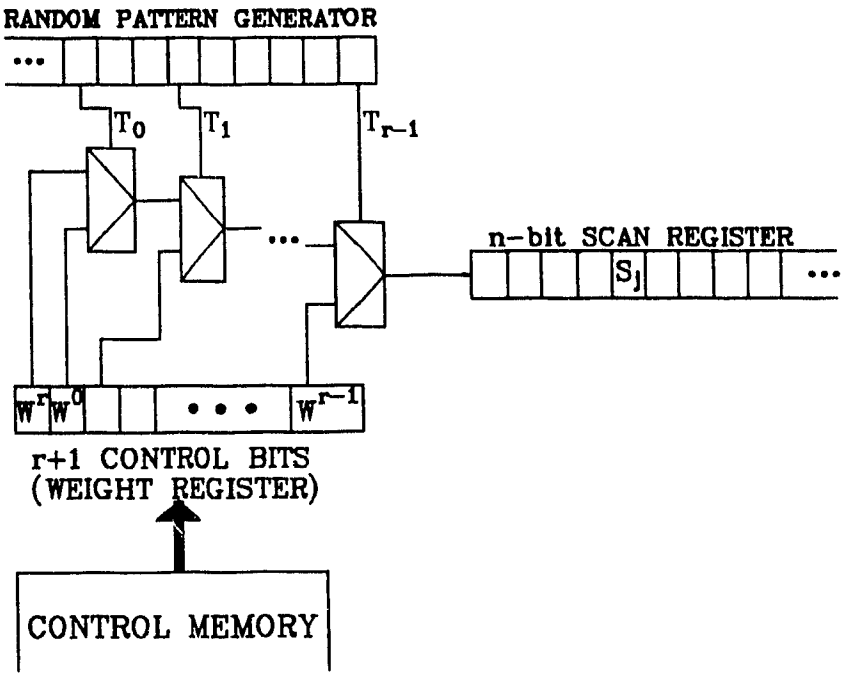


Figure 5.7 Canonical Weighting Circuit

Chapter 6 Weighted Random Built-In Self-Test

This chapter examines a BIST implementation of a WRP generator for full serial scan-path (hundreds to thousands of scan cells) circuits. Previously it has been suggested [Wun88] [Wai88] [Lis87] that multiple weight sets can be used to minimize the overall test length, however in a BIST environment, the high cost of implementing different weight distributions must be considered. For this reason, test pattern generation is performed using only one uniform random weight set and one non-uniform weight set of fairly coarse resolution. Experimental results will demonstrate that the performance of such a strategy is orders of magnitude better than uniform random pattern testing alone

The self-test methodology involves serially scanning in a test pattern and scanning out the corresponding test response into a compaction unit. Using such an approach makes the suggested BIST solution extendible to external testing. Experiments are performed using the single stuck-at fault model, however, the analysis can be applied to any fault model which does not require a specific ordering of input test patterns (e.g non-feedback bridging faults, multiple stuck-at, etc.)

The text is divided into two sections: first, an ad-hoc simulation based method of determining the characteristic weight sets is outlined; and second, a distributed WRP generator is proposed for local generation of the dually weighted sequence. This generator design is intended for BIST and is suitable for automated design. At various

stages, performance results are given in order to help demonstrate the effectiveness of the procedure.

6.1 Weight Set Estimation

Much work has been done concerning the off-line development of compact test sets and sequences (chapter 2), but on-chip test application schemes are rare (recall that on-chip storage requirements of even a fraction of the test set may be too large [Bas80] and unrealistic test lengths may be incurred when using only uniform random pattern genertion). The test pattern generation scheme discussed here is geared for efficient (in terms of time, storage and implementation) test application. It can be considered a means of transforming a conventional randomly generated test sequence into a pair of relatively short test sequences, one of which is non-uniformly weighted. Conceptually, the approach is similar to the joint random pattern and stored pattern testing scheme which was described previously in section 4.1.2, except that in this case, stored pattern generation is replaced by WRP generation. The rest of this section focuses on the design of the non-uniform weight set.

Unlike much of the previous work (Chapter 5), the weight estimation procedure presented relies neither on formal signal probability calculations nor on explicit analysis of circuit structure. As mentioned in chapter 5, in some methods which depend on such details (eg. [Bar87][Eic87]), fanout may directly affect the accuracy of the final weight set because conflicting weight assignments may be assigned to separate circuit nets which converge at the same gate. Also, if the weighting algorithm is such that a detectability value for each potential fault contributes to the result, signal probability estimates for redundant fault sites may pollute the weight computation.

In the work at hand, the conjecture is that sufficient circuit information pertinent to designing a weight set is contained in a test set, and can be extracted using simulation techniques. In fact, since the weight estimation process is based on the circuit's sim

ulated behaviour during test mode, the circuit can be considered a functional "black box" during the procedure. Because fanout and redundant faults are not explicitly considered in a functional circuit representation, they should not directly contribute to the accuracy of the final weight set. The existence of fanout and redundancies do, as usual, impact the computational effort of simulation tasks performed.

As an aid to the ensuing discussion, a flowchart of the entire scheme is provided in figure 6.1. The alphabetic tags associated with the various steps are refered to in the subsection(s) dealing with that stage of the process.

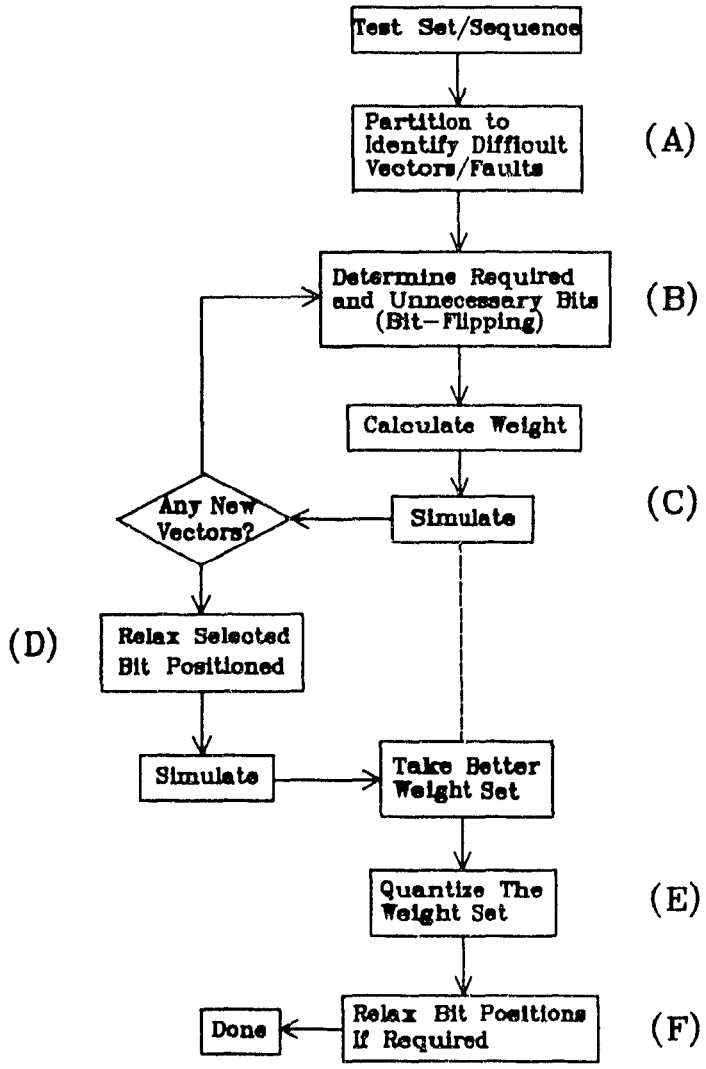


Figure 6.1 Flowchart of the Weight Estimation Procedure

The first step (A) is to identify the required test vectors which tend to extend the conventional random test length needed to generate a test set. This can be done by tracking an associated random pattern detection profile 6.2) and partitioning the curve at the point at which the rate of coverage begins to decrease by a user-defined threshold amount. Typically this might occur in the "knee" region of the curve. The groups of vectors found on either side of the partition point form two subsets, each of which defines a different weight.

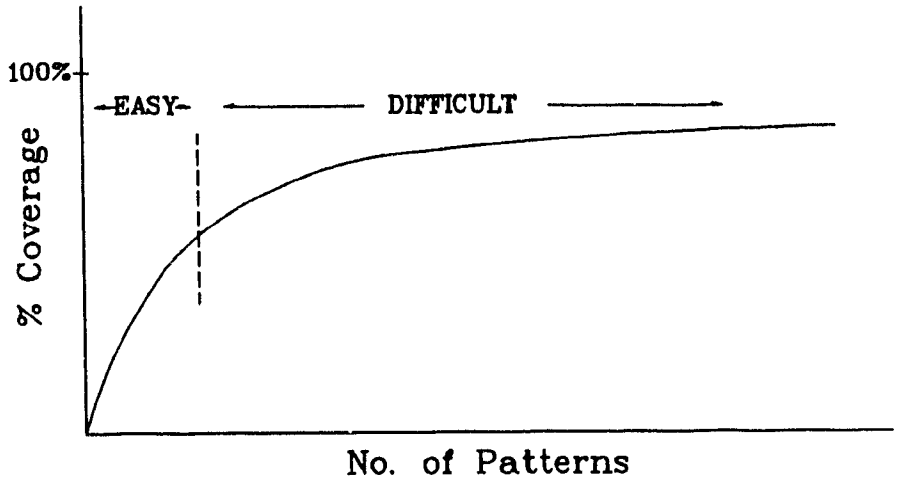


Figure 6.2 Progression of Coverage with Uniform Random Patterns

In the steeper sloped region to the left of the partition, faults classified as random **easy** are quickly detected in a relatively short test length. Since the performance of random patterns is already acceptable in this region, a uniform weight of 0.5 (random) is used to cover these faults. As the test length approaches infinity, random **difficult**[10] faults are found at a comparatively slower rate. This is characterised by the long flattened tail region to the right of the partition. Because the vectors occurring within this tail

10 Note that the qualitative classification of a fault as "difficult" or "easy" is always with respect to a specific type of generator, be it random or biased according to a specific weight set, unless specified as otherwise "difficult" will refer to uniform random generation

segment are more resistant[11] to random pattern generation, a weight set is calculated to cover them (and thus their associated faults) more rapidly. The expected effect of using such a pair of weights is shown in figure 6.3.

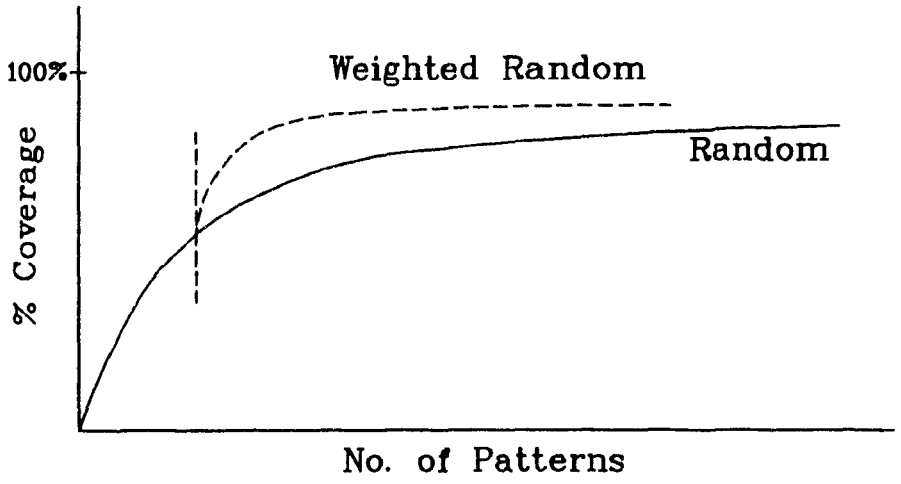


Figure 6.3 Expected effect of 2 weight sets

6.1.1 Processing the Tail Vectors (B)

In many cases it may be unrealistic to perform fault simulation until all difficult faults are detected. Instead, a cut-off limit may be imposed on the rate of coverage, after which simulation is terminated. In such an instance, although it is possible that not all difficult faults are covered by the test sequence, if the rate of coverage at the cut-off is chosen to be sufficiently small so that most of the difficult faults are found, the tail vectors can still be used to determine an initial non-uniform weight set. Since, aside from the circuit itself, the vectors for the detected difficult faults represent the only available information for determining the weight set, the estimated weight set will be geared to cover the detected difficult faults (called *target* faults). ATPG tools or a method such as that described in section 6.6 can be used to determine supplemental tail vectors for missed faults.

[11] The degree of a fault's "resistance" depends on the threshold criteria, thus the position of the partition

The pool of vectors which detect target faults is processed to provide the necessary information needed to calculate the non-uniform weight. Each test vector contains information concerning the circuit faults which it detects. Specifically, a subset of the bit assignments within a vector represents a bit ordering which detects the associated faults. However, some of this information may be redundant since some faults are repeatedly detected by subsequent tail vectors.

In order to formulate a weight set characteristic of the target faults, an attempt is made to reduce, if not remove, any redundant information contained within each tail end vector and isolate the bit assignments useful in detecting faults. The first task is accomplished by using a fault dropping scheme, demonstrated in figure 6.1, to assign to each tail vector (V) a list of unique target faults (t) which excludes those faults detected by any previous vector.

Vector	Original Fault List	Assigned Fault List
V_1	$t_1\ t_2\ t_3\ t_6$	$t_1\ t_2\ t_3\ t_6$
V_2	$t_1\ t_3\ t_4$	t_4
V_3	$t_1\ t_2\ t_3\ t_5$	t_5

Table 6.1 Reducing the contribution of multiple detection of faults

For each tail vector, V_i, not all bit assignments are instrumental in detecting its associated faults. Some simulation tools [Maa88] can be used to provide a partial estimate of the necessary bits per vector. ATPG tools can also be configured to provide some of this information.

In the experiments performed a ***bit flipping*** algorithm was used to determine the needed bits per vector. This is introduced here:

- Given an n bit tail vector V_j, create n new unique vectors $(v_1, .., v_i, .., v_n)$ each of whose Hamming distance from V_j is 1. This is done by inverting the ith bit of V_j to create v_i.

- Simulate each vector v_i with respect to the target faults associated with V_j.
  - if the coverage decreases then bit i is needed
    else bit i is marked "don't care" (unnecessary)

The "filtering" process described above results in a sparse test set in which only the needed bit assignments per tail vector are retained. Using this test set representation, a weight set which can cover the set of target faults can be estimated by calculating the ratio of the number of 1s to 0s within each bit position (input position). A generator defined as such, attempts to produce a test sequence containing vectors similar to (ideally identical to) that of the filtered test set

Vector	Before Bit Flipping						After Bit Flipping					
V_1	1	1	1	0	0	1	x	1	1	x	x	x
V_2	0	1	1	1	0	1	0	1	x	1	x	x
V_3	1	1	0	0	0	0	1	1	x	x	x	0
V_4	0	1	0	0	0	1	0	1	0	x	x	x
weight	.5	1	.5	.25	0	.75	.33	1	.5	1	.5	0

Table 6.2 Weight Set Estimation

In table 6.2, a weight set is calculated for a sample set of vectors before and after extraction of the needed bits. Notice that that the removal of the unnecessary bits (shown as 'x' after bit flipping) results in a much different weight for the first, fourth, fifth and sixth bit positions. Before bit flipping is performed, the definite assignments to the unnecessary bit positions act as "noise" which corrupt the weight estimate. This is apparent, for example, in the sixth bit position – before bit flipping the weight is 0.75 but it is found that a 0 bias is more appropriate. Filtering off these assignments should result in a more accurate weight set.

6.1.2 Results I

The test generation scheme is evaluated using five scannable circuits of varying size[12] The complexity of their topology is indicated in table 6.3.

NETLIST	#Faults	# Inputs	# Outputs	# Gates	# Levels
C6941	5673	247	250	3780	55
C9389	8075	725	786	5044	53
C11657	9836	615	687	6541	72
C30989	28572	1668	1668	16299	44
C35002	32985	1464	1578	16820	45

Table 6.3 General Circuit Information

The first stage of the process is to determine an initial test set and isolate the **difficult** faults. This is done by simulating[13], with a collapsed fault set, the order of 1 million random patterns and partitioning the sequence at the point where less than roughly 30 faults were detected in a window of 3000 consecutive random patterns[14] As mentioned before, the partition threshold is flexible. The results are shown in table 6.4 For each circuit, two test lengths and their respective fault coverages in terms of the number of irredundant faults left undetected are given. First, in column II, the test length required to reach the partition point (i.e. that covers the **easy** faults) is shown. Column III lists the fault coverage of this short sequence. The maximum number of random patterns simulated (L_o) and its associated coverage are given in columns IV and V respectively Finally, the number of **target difficult** faults identified in the test sequence after the

---

12 These were provided by Bell Northern Research

13 The fast fault simulator used is Tulip [Maa88] and redundancies were later found using [Cox90]

14 Although the initial test set was determined using simulation, it is stressed that it is not important how the test set was developed

I NETLIST	II R.P. Test Length	III # Undet Faults	IV Max. R.P. Test Length (L_o)	V # Undet Faults	VI # Target Faults (V - III)
C6941	5 5K	751	3M	7	744
C9389	12K	508	0 2M	0	508
C11657	5 3K	517	1M	94	423
C30989	13K	1954	2M	41	1913
C35002	15K	370	1M	33	337

Table 6.4 Initial (random pattern) Circuit Data

partition point (i.e after the test length of column II) is shown in column VI. This fault pool provides information used in the bit-flipping process.

The next step is to isolate the **difficult** vectors and extract the needed bits. Using this information, a weight set is then calculated and a set of 40K WRPs is simulated (table 6.5). This limit on the WRP test length is chosen to satisfy approximate test time restrictions[15] but in practice, of course, the upper bound can vary Column III (L_{wrp}) of table 6.5 is the point in this WRP test sequence after which coverage of the **difficult** faults does not increase significantly L_{rp} (col. II) is the number of extra random patterns required to cover the easy faults not yet detected. The coverage of the pair of weight sets is indicated in column IV by the number of faults left undetected after applying $L_{wrp} + L_{rp}$.

Test length improvements ranging from 1 to 3 orders of magnitude (i e powers of 10) are recorded. For example, more than 2 million pseudorandom patterns were needed to test C30989 with a coverage of 41 undetected faults, while 31 undetected faults remain after a joint sequence of 53K patterns Also, although a few target faults are missed, a number of previously undetected ones are found (col V) Recall that these newly detected faults could have required a number of random patterns far in excess of the

[15] Another measure of this would be to more specifically consider the scan chain length and limit the number of bits in the test sequence

I NETLIST	II L_{rp}	III L_{wrp}	IV # Undet. ($L_{wrp} + L_{rp}$)	V # New Found	VI # Targets Missed	VII # Undet ($L_{wrp} + L_o$)
C6941	5.5K	38K	**8**	7	8	0
C9389	12K	34K	**0**	0	0	0
C11657	5.3K	34K	**24**	77	7	17
C30989	13K	40K	**31**	33	23	8
C35002	15K	37K	**21**	22	10	12

Table 6.5 Initial WRP data

maximum amount originally simulated (table 6.4).

6.2 Improving the Estimated Weight Set (C,D)

Referring to figure 6.3, as expected, within the WRP region there is again an extended tail segment where WRP **difficult** faults are detected. This region may be excessively long because not all faults were detected in the original random pattern simulation, and/or characteristics of all faults are not sufficiently represented by one non-uniform weight set.

To overcome these problems, we propose two modifications to alter the weight set :

- Update the initial pool of target faults.
- Relax selected input biases.

These modifications are ad hoc and iterative in nature.

6.2.1 Phase 1 - Update Target Pool (C)

The effectiveness of the method depends on how well the existing pool of target faults represents all the existing **difficult** faults[16]. It is possible that simulation according to

[16] In the experiments performed, redundancies were not known initially

the the initial weight set can detect some extra faults which are not detectable using realistic random test lengths. These new test vectors are included into the target pool and the weight set is re-estimated. This process of iterative WRP generation and re-estimation of the weight set repeats until no new faults are detected.

6.2.1.1 Results II

The results of Phase 1 modification of the initial weight set is shown below in table 6.6. Here the number of *new faults* detected (col V) does not include those found in the initial WRP run (table 6.5). The total number of target faults is updated to the total number of difficult faults found thus far in the process.

I NETLIST	II L_{rp}	III L_{wrp}	IV # Undet. ($L_{wrp} + L_{rp}$)	V # New Found	VI # Target Missed	VII # Undet. ($L_{wrp} + L_{o}$
C6941	5.5K	27K	**5**	0	5	0
C9389	12K	34K	**0**	0	0	0
C11657	5.3K	22.2K	**7**	17	7	0
C30989	13K	40K	**31**	0	23	8
C35002	15K	29K	**12**	4	5	7

Table 6.6 Phase1 runs of 40K length

As expected, there is a general improvement in the weighted test length (col. III); in three cases (C6941, C11657 and C35002) accompanied by an increase in coverage The reduction of test length may in some cases seem small but depending on the number of input nodes, the impact on test application time can be quite significant

6.2.2 Phase 2 - Relax Pin Biases (D)

In an attempt to further reduce the weighted test length, the biases of selected pins are relaxed. After a suitable weight set is found in Phase 1, the circuit is re-simulated

using the weighted generator and only the target pool of faults is considered. Again, by noting the progression of coverage, there will be a small group of vectors which are found after a long WRP sequence This effect is expected since one bias is being used to characterize many possibly different faults. The assumption is that these few WRP **difficult** vectors strongly differ from the generator bias at several positions. A partial remedy is to force some of these conflicting pin biases to 50% This compromise should not adversely affect the test length if the number of positions modified is small

The method of selecting bit positions to be relaxed is rather straightforward According to a threshold criteria, vectors found in the end of the WRP tail region are stripped from the WRP test set. An example of such a threshold is to strip all vectors found after the point where no fault is detected by 1000 consecutive WRPs

Using the algorithm of section 6.1, a weight set is estimated from the stripped WRP vectors and is compared to the generator bias. Bit positions which differ by a user-defined amount (e.g. 0 7), are relaxed. For example :

Generator Weight	1	0	.2	.7	**.8**	**.2**	.1
Tail Weight	1	0	.4	.8	**.1**	**.9**	.3
New Weight	1	0	2	7	**.5**	**.5**	1

Note that this is a potentially volatile step because disturbing the generator bias may negatively affect the detection of WRP **easy** faults. As a result, it is recommended that this phase be performed only once and the number of positions relaxed be small.

6.2.2.1 Results III

Table 6 7 is the results after phase 2 modification.

Aside from c9389, there is little difference in the WRP test lengths of this and the phase 1 run. In 3 cases though, the coverage increased slightly. Also, notice that for C30989, L_{wrp} is the 40K limit. If ten thousand more weighted patterns are simulated,

I NETLIST	II L_{rp}	III L_{wrp}	IV # Undet. $(L_{rp} + L_{wrp})$	V # New Found	VI # Target Missed	VII # Bits Relaxed
C6941	5.5K	33K	**4**	0	4	5
C9389	12K	4K	**0**	0	0	6
C11657	5 3K	25K	**0**	0	0	10
C30989	13K	40K	**30**	0	22	16
C35002	15K	29K	**12**	0	5	6

Table 6.7 Refined WRP Data

the coverage is 22 undetected faults, thus illustrating the possible tradeoff between test time and coverage. Relaxing an input's bias to 0.5 also reduces the implementation overhead since a modified scan cell is not needed at that position.

In all the cases, because test length did not suffer, it could be beneficial to investigate the maximum number of inputs which could be relaxed without seriously increasing the WRP test length and coverage

A summary of the performance of the complete generation scheme in reducing the random pattern test length otherwise needed to achieve equivalent coverage, is shown in table 6 8. It was mentioned earilier that weighted generation can detect faults which would require a number of random patterns far greater than the amount originally simulated (L_o). If this trend persists in other circuits, the conventional 2 part test generation strategy of using random vectors followed by DTPG can be modified to include a weighted random stage after the random pattern step. The results of such a process is shown in column VII$(L_{wrp} + L_o)$ of tables 6.5 and 6 6

6.3 Hardware Implementation

One of the major concerns when building an on-chip generator is to minimize the area overhead without severely compromising test quality and testability of the BIST cir-

NETLIST	Refined WRP Data L_{rp}	L_{wrp}	# Undet Faults	Equiv Rand Pat Test Len	Test Length Reduction
C6941	5.5K	33K	**4**	4M	99%
C9389	12K	4K	**0**	0.5M	92%
C11657	5.3K	25K	**0**	30M	99%
C30989	13K	40K	**30**	4M	98%
C35002	15K	29K	**12**	>5M	99%

Table 6.8 Weighted Testing v s Random Testing

cuitry. Two interdependent issues are involved in the proposed implementation strategy :

- The hardware used to generate patterns according to a pair of weight sets must be simple and transparent to the CUT It must also be general enough to be used in any scan circuit testable with one non-uniform weight
- The weight set determined must be adapted to suit the generator

The suggested approach is to redesign specific scan cells so that from the pattern generation point of view, the CUT appears testable with a uniformly distributed pseudorandom sequence.

6.3.1 Local Generation of Dually Weighted Test Patterns

A Distributed BIST Generator provides test vectors according to **both** the equiprobable and determined weight sets. It consists of a generating scan chain (GSCAN) and an on-chip autonomous random pattern generator such as a CA or LFSR The global effect is that each generator output (CUT input) can be made to be dually weighted – generating patterns according to the bias of the source generator, or according to a hard-wired weight (figure 6.4)

GSCAN is an altered scan chain which serves (apart from its other functions) as an interface between a random pattern generator and a WRP testable circuit At a number

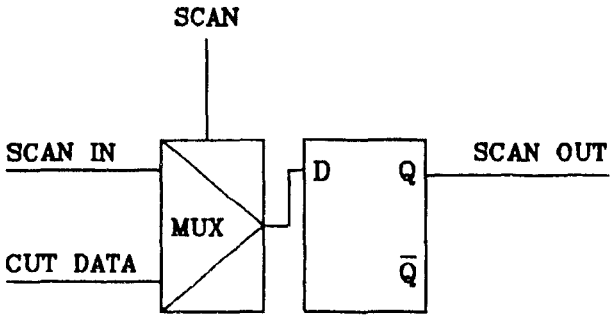


Figure 6.5 Generic Scan cell

of times proportional to the ratio of the WRP test length to the random test length, the random input at selected pin positions is locally converted to a weighted circuit inputs of specific biases This is done by replacing standard scan elements by modified blocks which perform the conversion. These new cells are themselves generic and can be included in a CAD library for automated design and layout.

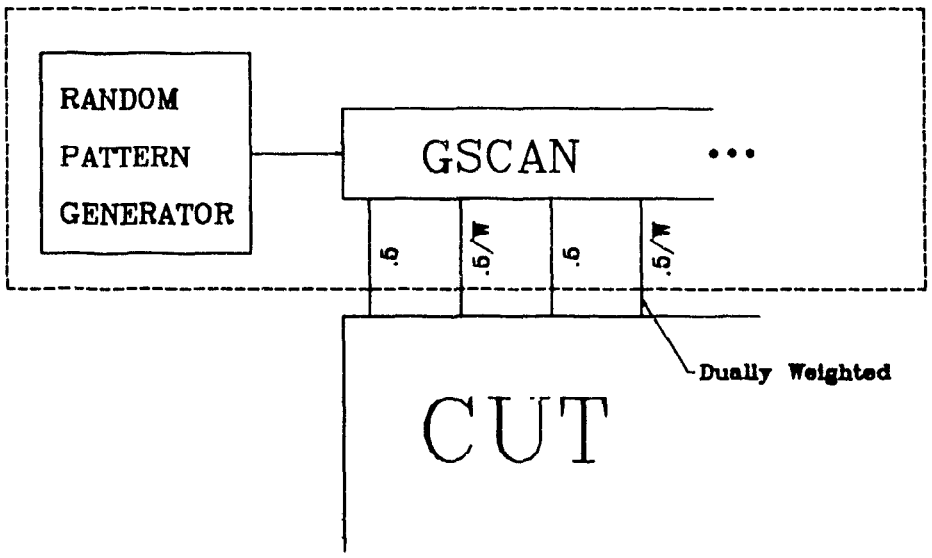


Figure 6.4 Distributed Generation of Patterns according to 2 weight sets

Since the weight set is a one to one mapping of bias values to input pins, the scan cells to be replaced are those which correspond to a bias other than 0.5.

6.3.2 The Circuit

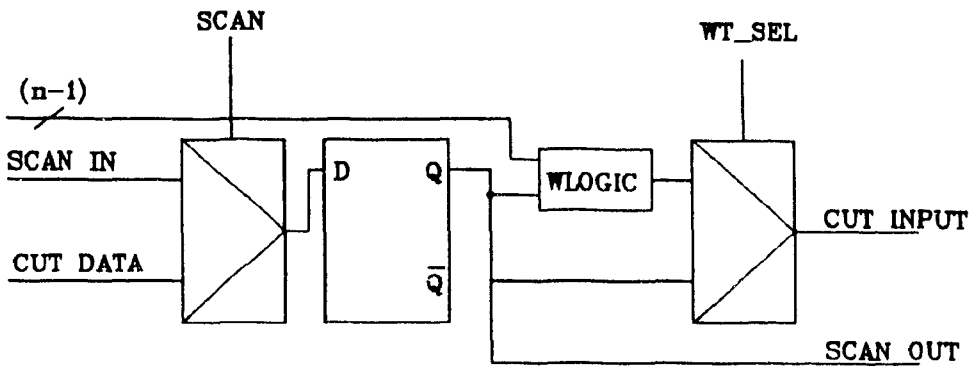


Figure 6.6 Scan cell modified for WRP generation

The logical block diagram of the circuit for locally generating a biased input sequence, is shown in figure 6.6. It differs from a regular scan element (figure 6.5) in that there are :

- n-1 possible extra input lines from **previous** scan elements.
- Extra combinational logic (WLOGIC) to generate the biased stream.
- Output multiplexing to choose between biased streams
- Independence between the scan output and the input to the CUT
- An extra control line (WT-SEL) required globally

The means of generating a weighted sequence exploits the output signal probability of logic gates. Extra combinational logic (shown as WLOGIC in figure 6.6) combines n random input streams, one from the resident flip flop and $n-1$ from the $n-1$ previous cells, to give a weighted output stream For example if WLOGIC is an AND gate and n is two, the bias of the weighted sequence is 0.25.

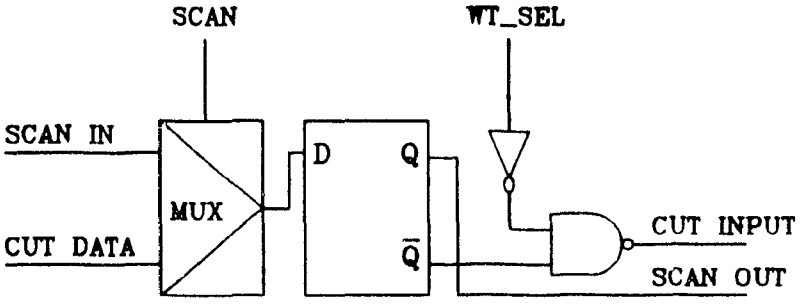


Figure 6.7 GSCAN cell for a bias of 1

In general, the output multiplexer and WLOGIC can be represented by a small, functionally equivalent, combinational circuit. Sample GSCAN cell designs which conditionally generate bit streams of bias 1, 0, 0.25 and 0.75 are shown in figures 6.7 to 6.10. More finely tuned weights such as 0.125 and 0.875, etc., can be generated if values from greater than 1 neighbour are combined (i.e. $n \geq 3$). In such cases, however, the cost of routing might become prohibitive.

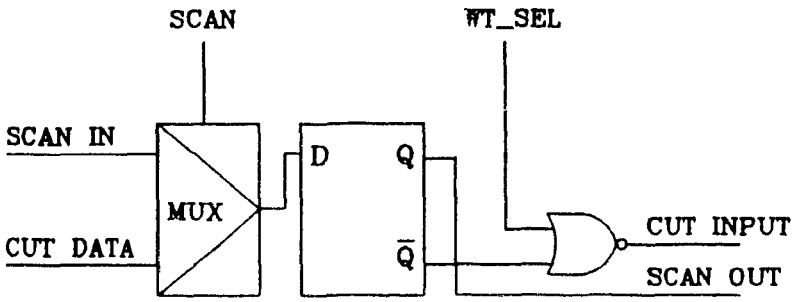


Figure 6.8 GSCAN cell for a bias of 0

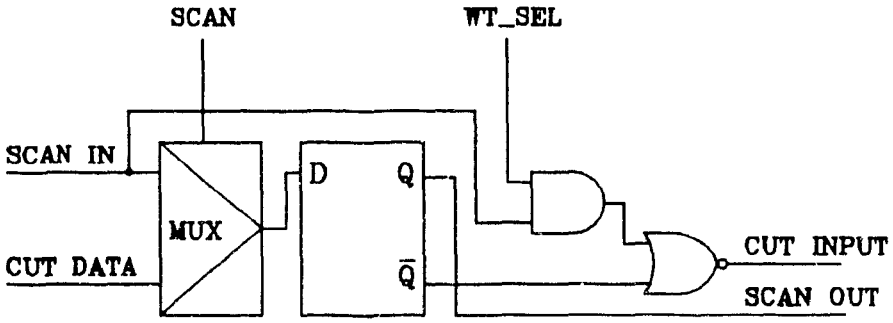


Figure 6.9 GSCAN cell for a bias of 0.25

6.3.3 Mixed Generation/Application of Uniform and Weighted Patterns

WT-SEL is a control line used to select between the weighted bit stream generated locally and the contents of a GSCAN cell's resident D-flip flop (D-FF). The signal can be applied externally or, in a true BIST sense, generated by logic connected to the RPG. Furthermore, the signal can be encoded such that weighted and non-weighted test

In general, the output multiplexer and WLOGIC can be represented by a small, functionally equivalent, combinational circuit. Sample GSCAN cell designs which conditionally generate bit streams of bias 1, 0, 0.25 and 0.75 are shown in figures 6.7 to 6.10. More finely tuned weights such as 0.125 and 0.875, etc., can be generated if values from greater than 1 neighbour are combined (i.e. $n \geq 3$). In such cases, however, the cost of routing might become prohibitive.

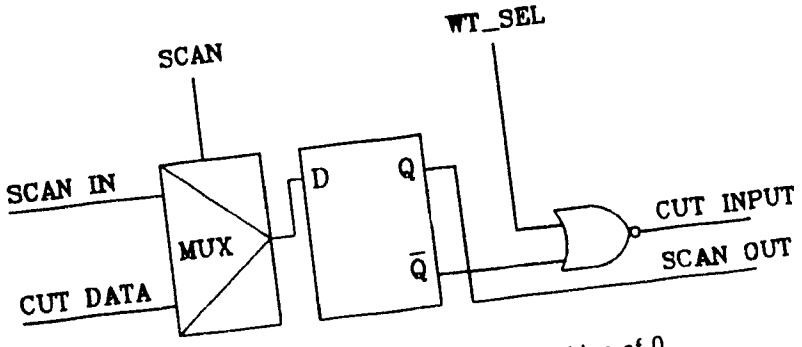


Figure 6.8 GSCAN cell for a bias of 0

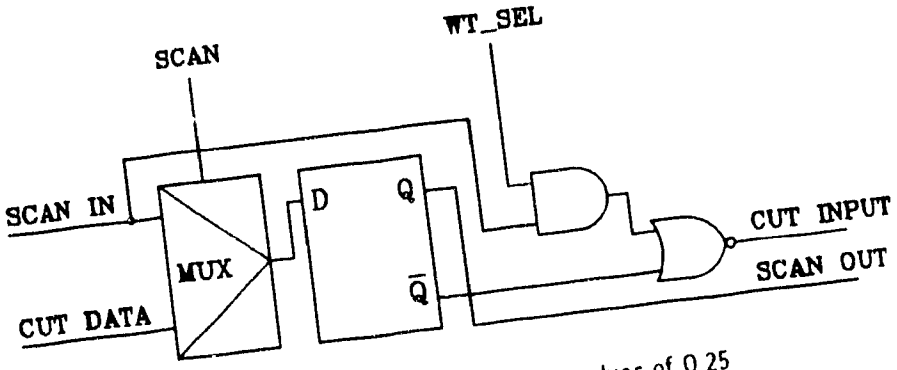


Figure 6.9 GSCAN cell for a bias of 0.25

6.3.3 Mixed Generation/Application of Uniform and Weighted Patterns

WT-SEL is a control line used to select between the weighted bit stream generated locally and the contents of a GSCAN cell's resident D-flip flop (D-FF). The signal can be applied externally or, in a true BIST sense, generated by logic connected to the RPG. Furthermore, the signal can be encoded such that weighted and non-weighted test

During TEST, WT-SEL is itself a weighted sequence whose weight reflects the proportional sizes of the WRP and random pattern test lengths. It is used to choose either the biased stream generated or the random sequence scanned in from the RPG. This technique is possible because the order in which the individual test vectors (weighted vs. random) are applied is not critical when testing combinational circuits, using the single stuck-at model.

0 - Mode

In order to arbitrarily select the D-FF line, the value of WT-SEL can be forced to 0 (or 1)

It is a simple task to integrate other modes of operation to satisfy other desired scan functions.

6.3.4 Area Penalty

Compared to a regular scan element, the area overhead of a redesigned scan cell in which the weighted stream is 0.25, 0.75 or 1.0 is :

- a pair of 2 input gates.
- the extra routing since the scan output is independent of the CUT input.

In the case of a 0 bias, a simple NOR (4 transistors) is needed. If pin biases are chosen such that inputs are needed from greater than 1 neighbour, the overhead and layout complexity increases due to the routing from progressively distant cells

With reference to a conventional random pattern BIST implementation, the extra area penalty of this scheme is due to the contribution of each modified scan cell, the logic needed to generate WT-SEL and the additional interconnect required to globally distribute this control signal. More specific results concerning the area overhead of the five circuits under consideration are given in section 6.3 5.1

6.3.5 Quantization of Pin Biases (E).

Thresholds are established to force the individual biases of the estimated weight set to implementable values. The criteria should ensure that:

- The overall features of the experimental bias, especially the extreme values, are retained.
- The levels are chosen to facilitate minimum area overhead.

The latter point implies a tradeoff between design complexity, fault coverage and test time. For example, the most attractive levels in terms of area and routing are 0, 0.25, 0.5, 0.75 and 1. However there is a potential for loss when only these values are used because extreme pin biases, such as 0.05 and 0.95, are forced to 0.25 and 0.75 respectively. This will, in most cases, result in an increase in test length or a possible reduction of coverage if the test size is limited. One solution, of course, is to use more quantization levels but this results in higher overhead.

In general, the thresholds and number of quantization levels used depend on what is deemed acceptable production-wise. It will be shown experimentally in the next section that biases restricted to 0, 0.25, 0.5 0.75 and 1 are indeed sufficient to attain acceptable fault coverage. The results of including levels 0.125 and 0.875 are also included for completeness.

6.3.5.1 Results IV

Thresholds are defined to allocate ranges of biases to fixed values. The conditions used in the ensuing discussion are shown in figure 6.12.

For each test circuit the quantization effects on the weight sets found in phase 1 and phase 2 (after pin relaxing) are examined. The discussion is limited to case 1 and case 4 thresholds but more detailed information concerning all runs is given in the appendix.

Table 6.10 are the results of using the bias levels {0, 0.125, 0.25, 0.5, 0.75, 0.875, 1

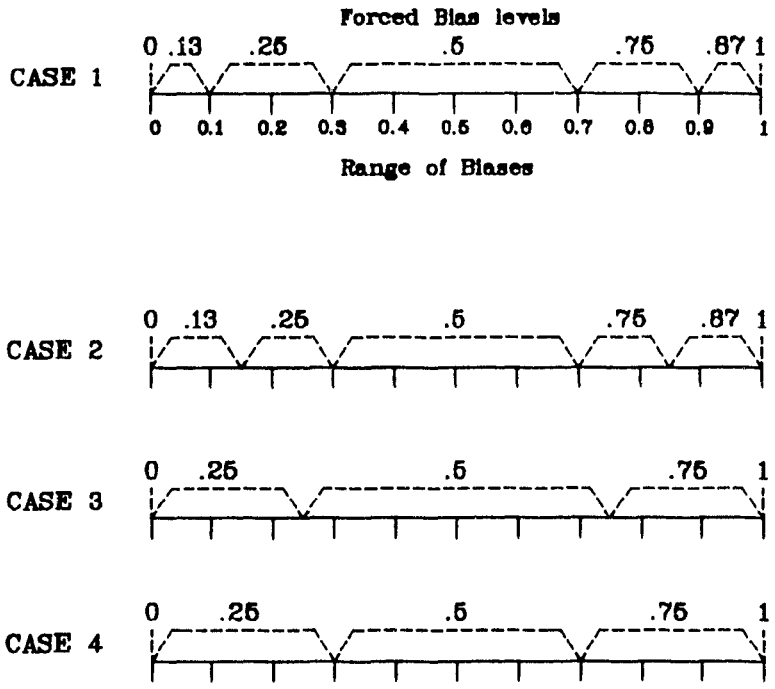


Figure 6.12 Thresholding conditions for Quantization Runs

CASE 1	Before Pin Relaxing			After Pin Relaxing		
NETLIST	L_{rp}	L_{wrp}	# Undet. Faults	L_{rp}	L_{wrp}	# Undet. Faults
C6941	0.5K	35K	9	0.5K	40K	7
C9389	11.5K	20.5K	0	11.5K	6K	0
C11657	5.3K	27K	9	5.3K	37 5K	4
C30989	13.5K	38.5K	56	13.5K	39K	54
C35002	15K	38.4K	17	15K	38.4K	17

Table 6.10 Quantization Runs for Case 1 – 7 bias levels

and table 6.11 uses the levels <0, 0.25, 0.5, 0.75, 1>.

The impact of input pin relaxing (phase 2), in terms of coverage, is mixed. This in itself is an interesting result When a decrease in coverage due to pin relaxing does occur, it is very small but is accompanied by a reduction in the number of scan cells to be replaced. Thus there is possible merit to phase 2 modifications. Furthermore, it

CASE 4	Before Pin Relaxing			After Pin Relaxing		
NETLIST	L_{rp}	L_{wrp}	# Undet. Faults	L_{rp}	L_{wrp}	# Undet. Faults
C6941	0.4K	39K	36	0.4K	39K	40
C9389	11.5K	20.5K	0	11.5K	6K	0
C11657	5.3K	28K	7	5.3K	36K	8
C30989	11.5K	39K	73	11.5K	39.5K	73
C35002	15K	30.8K	25	15K	30.8K	25

Table 6.11 Quantization Runs for CASE 4 – 5 bias levels

is incorrect to assume that quantization performs the same degree of pin relaxing as phase 2. For example, using case 1 thresholding, a pin with bias 0.10 will be forced to 0.125 whereas phase 2 operations may force it to 0.5.

The result of using a joint scheme with 7 bias levels, in terms of both coverage and test length, is much better than that of uniform random generation alone. In a more practical sense, the coarser weight set of 5 levels uses only the less complex GSCAN cells and also produces a significant improvement. The implementation overhead (beyond scan) for each circuit of this latter case is approximately 3%. This is calculated by using an industrial standard cell library to estimate the size of each circuit and to this amount, the extra area needed in order to modify the scan chain is added. The distribution of pin biases and the size overheads cited in section 6.3.4 are used to approximate the additional area. Information concerning input distributions and additional runs can be found in the appendix.

6.3.6 LFSR-Based Scheme

In the previous experimental analysis, software random pattern generation (therefore almost no correlation between generated input bits) was used to evaluate the potential of a dually weighted generation scheme and examine the effect of using a coarse weight set. In order to evaluate the **mixed-weighted** implementation scheme, this section presents

the results of simulating a model of the distributed test structure using a 32-bit LFSR (table 6.12) and 5 bias levels. The performance is compared to that of 'ideal' uniform random pattern testing implemented in software.

For each circuit a 70 thousand mixed-weighted pattern test sequence (Test Len.) is generated with a weighted contribution indicated by the bias of the WT-SEL signal. The number of undetected faults at the 50 thousand and 70 thousand mark is shown along with the random pattern (R P.) test length needed for equivalent coverage

NETLIST	WT_SEL_{bias}	Mixed-Weighted Test Len.	# Undet Faults	Equiv R.P. Test Len.
C6941	75%	50K	19	1 9M
		70K	15	1 9M
C9389	13%	50K	4	120K
		70K	1	125K
C11657	75%	50K	23	15M
		70K	13	>30M
C30989	63%	50K	95	1M
		70K	55	1 7M
		150K	23	5M
		200K	18	>5M
C35002	37%	50K	58	144K
		70K	28	2 5M

Table 6.12 Comparison of LFSR-based Mixed-Weighted Scheme and Random Pattern Testing

Higher fault coverage is expected if the entire analysis (partitioning fault sets etc.) was performed using an LFSR. It has also been found from experience that changing the LFSR seed during test mode positively affects the rate of coverage.

Graphs of the progression of coverage of both the uniform random and the LFSR-based **mixed-weighted** testing techniques are given for all 5 circuits in figure 6.13 to 6.17 The test lengths are plotted on a log scale and the fault coverage is the percentage of all irredundant faults which are detected.

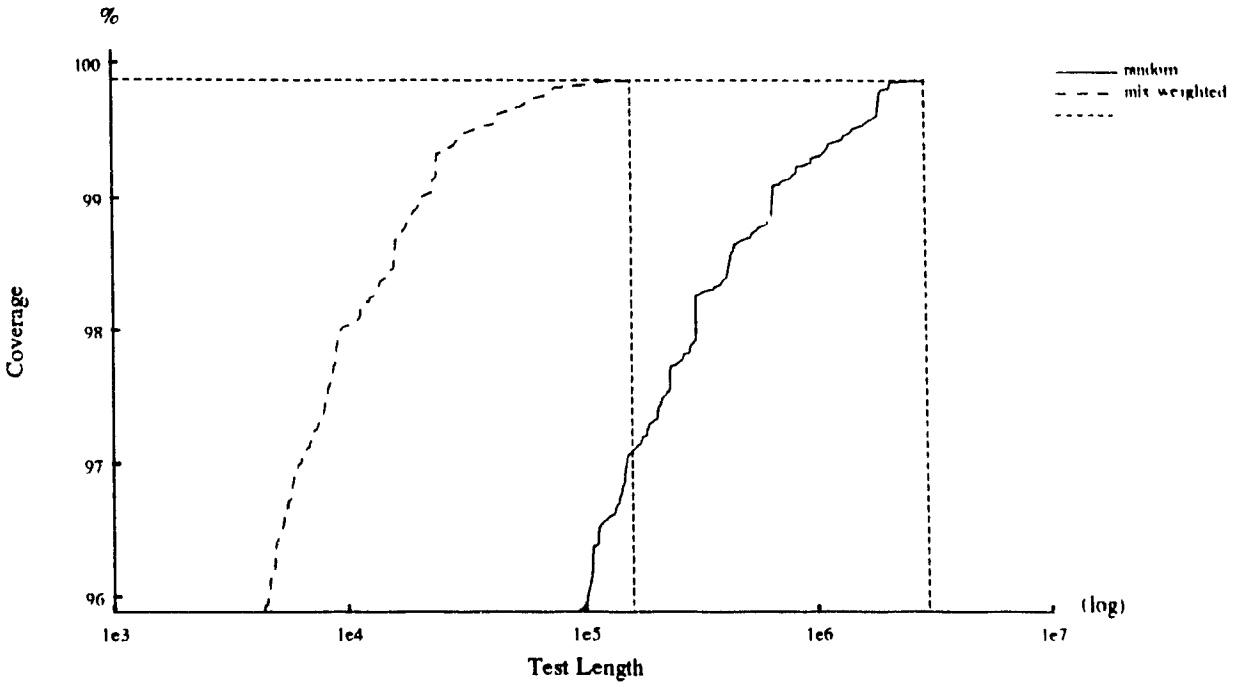


Figure 6.13 Coverage Plots for C6941

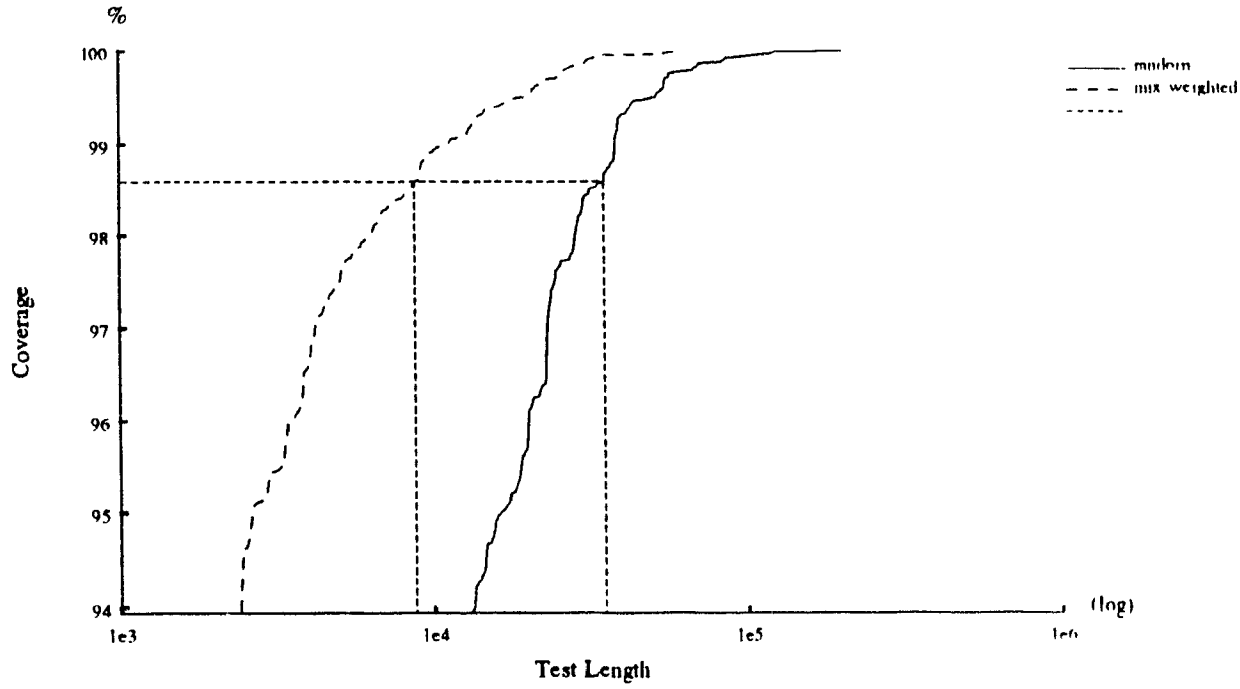


Figure 6.14 Coverage Plots for C9389

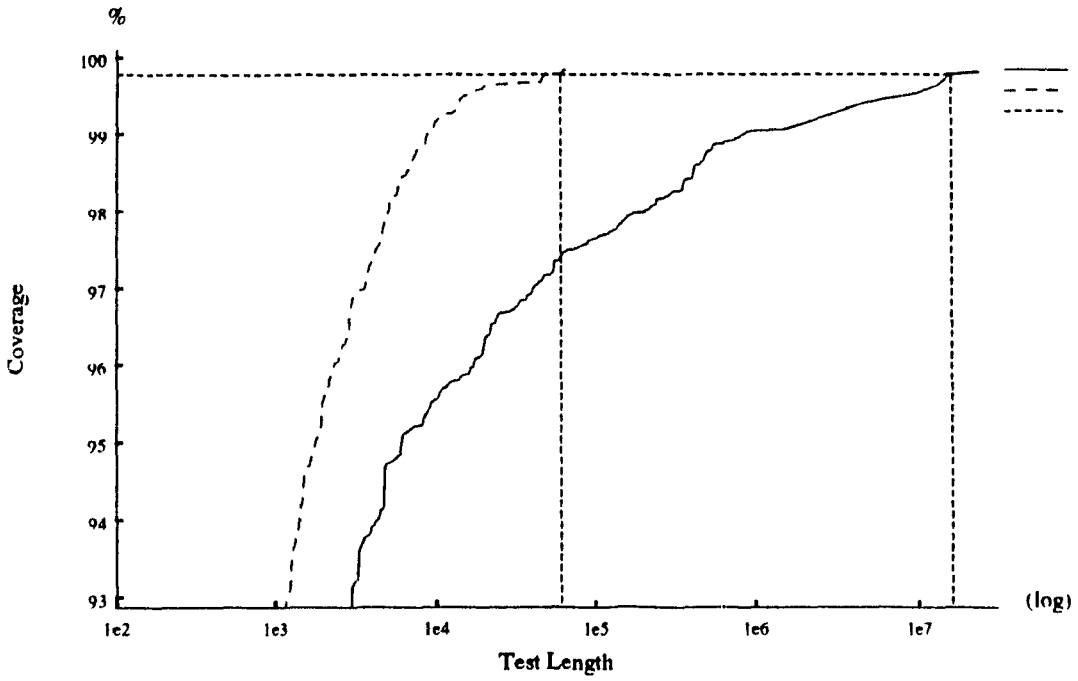


Figure 6.15 Coverage Plots for C11657

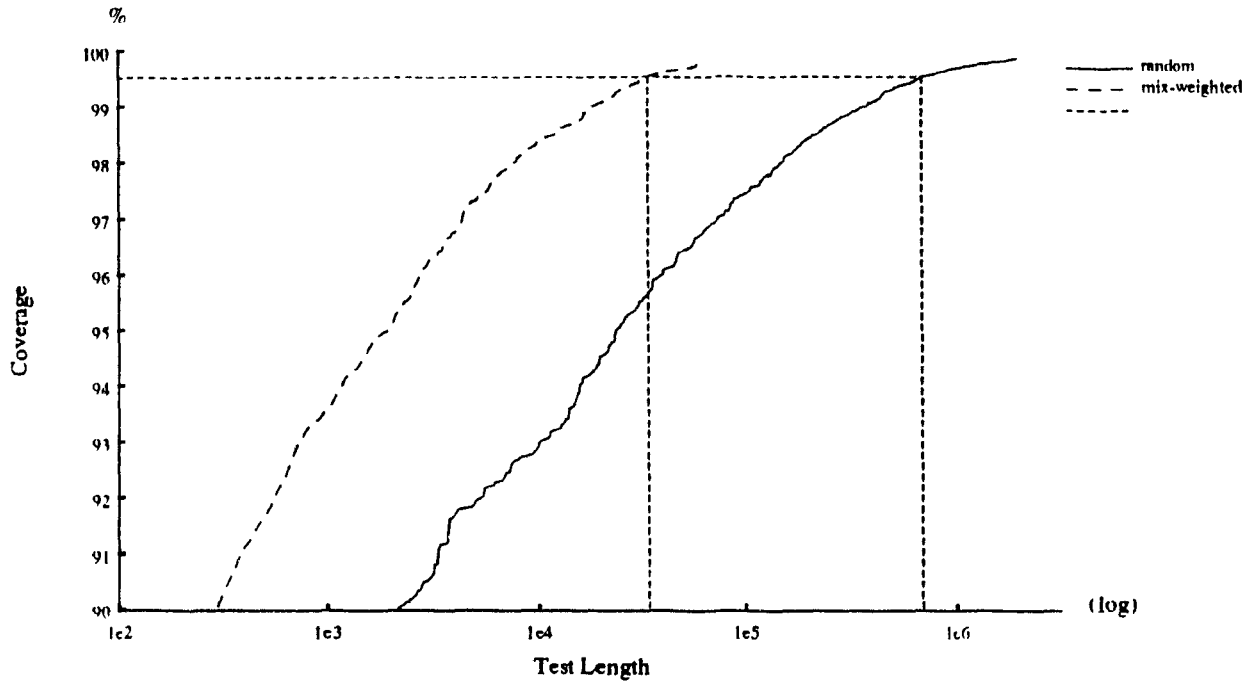


Figure 6.16 Coverage Plots for C30989

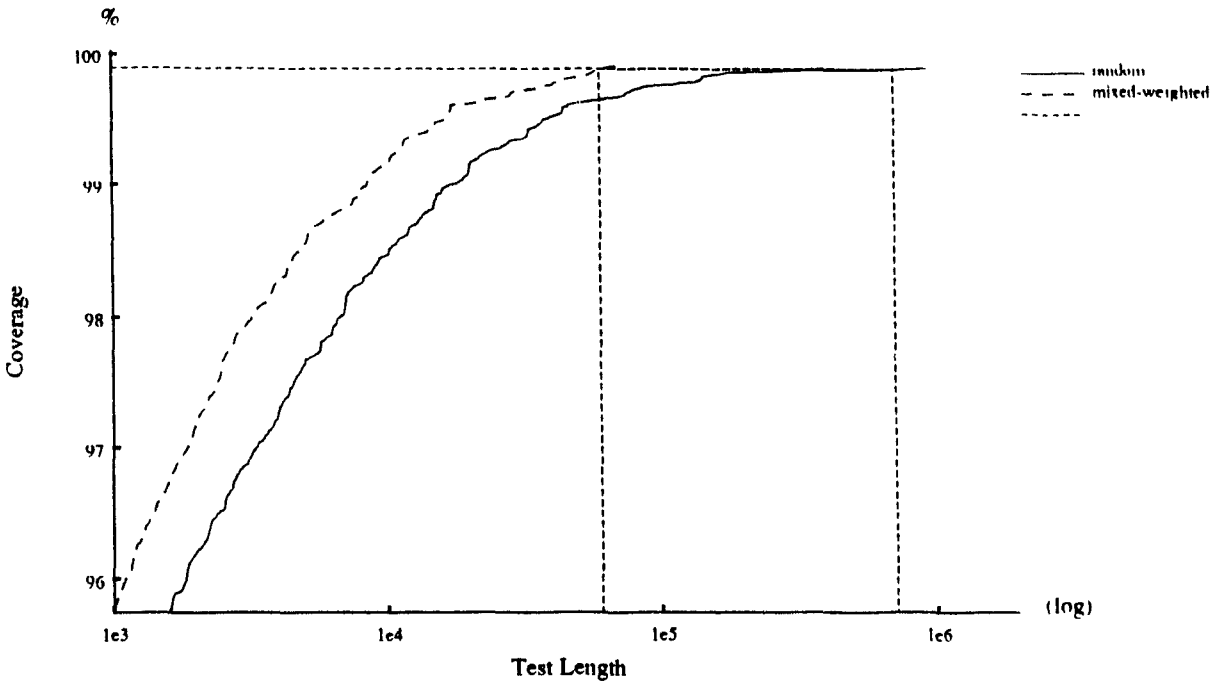


Figure 6.17 Coverage Plots for C35002

For these cases it is shown that random pattern test length can be reduced by orders of magnitude using the proposed mixed-weighted generation technique.

6.4 Other Test Circuits

Recently, other large benchmark circuits became available [ISC89]. These sequential circuits are similar in structure to the five test circuits already considered and were adapted for experimentation by replacing flip-flop elements by input and output pins. The sizes of the converted structures are indicated by the number of lines (e.g. C9234 is a combinational circuit with 9234 lines). Experiments were performed to develop a non-quantized refined weight set. As in the previous test cases, no claim is made concerning the optimality of the weight however the intention is to significantly reduce the random test length needed for equivalent coverage.

Table 6.13 contains some basic circuit information. Here, the total number of irredun-

dant faults is in terms of a collapsed fault set, and the redundancies were identified[17] Analogous to table 6.4, table 6.14 is the initial random pattern data used to derive the weight sets, and in order to aid evaluation of the generation scheme the results of some extended random pattern simulations are given in table 6.15.

NETLIST	#Faults irredundant	# Faults redundant	# Inputs	# Outputs
C9234	6475	452	247	250
C13207	9664	151	700	790
C15850	11336	389	611	684
C38417	31015	165	1664	1742
C38584	34797	1506	1464	1730

Table 6.13 General Circuit Information

I NETLIST	II R.P. Test Length	III # Undet Faults	IV Max. R.P. Test Length (L_o)	V # Undet Faults	VI # Target Faults (V - III)
C9234	5.5K	781	3M	31	750
C13207	11.5K	589	0.6M	0	589
C15850	6K	682	1M	78	603
C38417	16K	1766	2M	32	1734
C38584	15K	338	2M	29	309

Table 6.14 Initial (random pattern) Circuit Data

Using the developed refined weight set, the performance of the complete generation scheme is outlined in table 6.16. Software random pattern generation is used for this evaluation. In each case a set of 200K WRPs are simulated. Shown for each circuit is the weighted test length (L_{wrp}) and corresponding fault coverage at the point at which

[17] the 452 faults for C9234 are not proven redundant at this time

NETLIST	L_0	#Undet. Faults
C9234	20M	3
C13207	0.6M	0
C15850	120M	0
C38417	10M	5
C38584	10M	17

Table 6.15 Extended Uniform Random Pattern Circuit Data

fault detection ceased. Also, in cases where this length is in excess of 40K (C9234, C38417 and C38584), the number yet undetected faults at the 40K mark is listed.

NETLIST	Refined WRP Data			Equiv Rand. Pat. Test Len	Test Length Reduction
	L_{rp}	L_{wrp}	# Undet Faults		
C9234	5.5K	40K	**31**	3M	98%
	5.5K	150K	**0**	>20M	>99%
C13207	11.5K	7.4K	**0**	0.6M	96%
C15850	6K	29K	**0**	120M	>99%
C38417	16K	40K	**17**	10M	99%
	16K	186K	**0**	>10M	>99%
C38584	15K	40K	**9**	>10M	>99%
	15K	77K	**3**	>10M	>99%

Table 6.16 Weighted Testing v s Random Testing

Again, it is seen that the proposed generation scheme dramatically reduces the otherwise needed random test length. In all cases the random pattern test length is reduced by more than 96 % for an equivalent level of coverage.

6.5 Comments on Overhead and Testability

6.5.1 Computation Overhead

As mentioned, the thrust of this work is to provide an easily generatable test sequence and a corresponding BIST implementation. Defining overhead as the extra amount of work required to realize the weighting algorithm, the effort involved in generating the initial test set is neglected since it must be constructed anyway. The overhead of the method as presented then, is due to only the bit flipping process and the extra simulation needed to verify and fine tune the determined weight set. Other tasks such as partitioning the test sequence and calculating the weight set from processed vectors require negligible effort. In terms of the amount of patterns simulated, the computation penalty is $O(nv) + O(m)$ simulated patterns where :

- n = number of bits per vector.
- v = number of vectors processed.
- m = imposed maximum number of vectors in the test sequence.
- $O(m) \approx$ rough contribution of simulation needed to verify and modify determined weight set (e.g. phase 1 and phase 2).

$O(nv)$ is the contribution of the bit-flipping process. The work involoved in computing the weight set from the set of processed vectors is neglegible. In the cases examined, v was roughly 200 but this value varies with the chosen partition threshold between easy and difficult faults. Also, as mentioned previously, not all bits need be flipped when the vectors are processed since some simulators and DTPG tools can provide a partial estimate of the required bits.

6.5.2 Accuracy of Extracted Data

Even with steps taken to reduce redundancy, the data extracted using the bit flipping algorithm may not be optimal. This is best demonstrated by an example

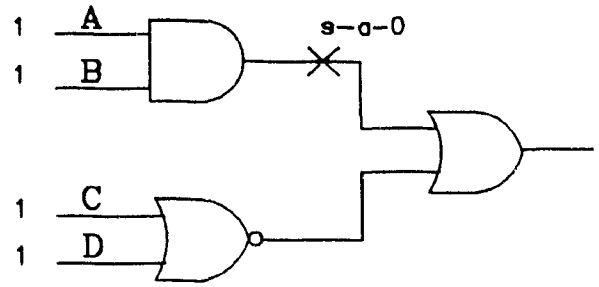


Figure 6.18 Sample Circuit for Bit Flipping

Figure 6.18 shows a simple 4 input circuit with test vector '1111' used to detect the illustrated stuck at 0 fault. Stepping through the bit flipping process (table 6.17), only the inputs to line A and B are tagged as needed. The resulting processed vector is '11XX' where 'X' denotes a discarded bit.

	A B C D	NEEDED
TEST VECTOR	1 1 1 1	
Bit1	0 1 1 1	√
Bit2	1 0 1 1	√
Bit3	1 1 0 1	
Bit4	1 1 1 0	
EXTRACTED VECTOR	1 1 x x	

Table 6.17 Bit flipping for sample circuit

By definition of a discarded bit, the specified fault should be detected regardless of the assignments to lines C and D. This is not true however, if C and D are assigned 00 (ie the test vector generated is '1100'). The experimental results show however, that this first order approximation of the needed bits is sufficient.

6.5.3 Testability of Modified Scan cells

In the implementation scheme of section 6.3, no redundant faults are introduced. This is demonstrated for figure 6.10 (0.75 bias) but a similar argument can be made for the

other modified scan cells In figure 6.19 it is assumed that line 4 (CUT INPUT) is fully testable. This is not an unreasonable assumption since this line already exists in the unmodified circuit. The problem of showing that lines 1,2,3 and 5 are testable reduces to propagating the effect of a fault on those lines to line 4

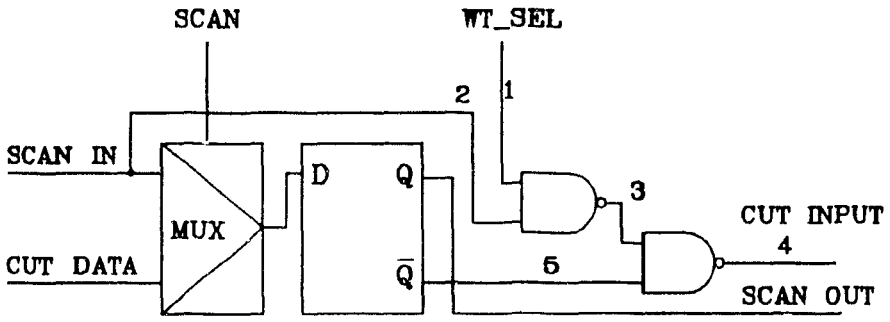


Figure 6.19 Introduced Fault Sites for a Modified Scan Cell

There are 8 possible single stuck-at faults on the target lines. Using a fault dropping scheme and recalling that line 4 is testable, four faults remain, each of which can be tested with the pattern(s) shown below in table 6.18 ·

Line	Fault	WT-SEL	SCAN-IN	$\overline{Q}$
1	s-a-1	0	1	1
2	s-a-1	1	0	1
3	s-a-1	1	1	1
5	s-a-1	0	0	0
		0	1	0
		1	0	0

Table 6.18 patterns

Of course, this only shows that the introduced circuit faults are not redundant, whether or not they are actually detected depends on the test length. If they are not detected within the specified test interval, a feedback loop could be added to the structure enabling the undetected fault(s) to be covered in the functional test of the scan chain.

Note that $\overline{Q}$ is used as input to the weighting logic in order to save a pair of transistors

If this were not done, tests for stuck faults on line 5 (now mux output Q) could be included in the functional test of the scan chain.

6.5.4 Possible Missing Input State (F).

By permuting the bits at neighbouring input sites to form weighted stimuli, it is expected that some correlation between neighbouring input bits is created during weighted pattern application.

Examples where this can occur are instances where a generic scan cell, at position i, is immediately followed by one modified to conditionally generate a bias of 0.25 or one modified to conditionally generate a bias of 0.75. The prior scenario is illustrated in figure 6.20.

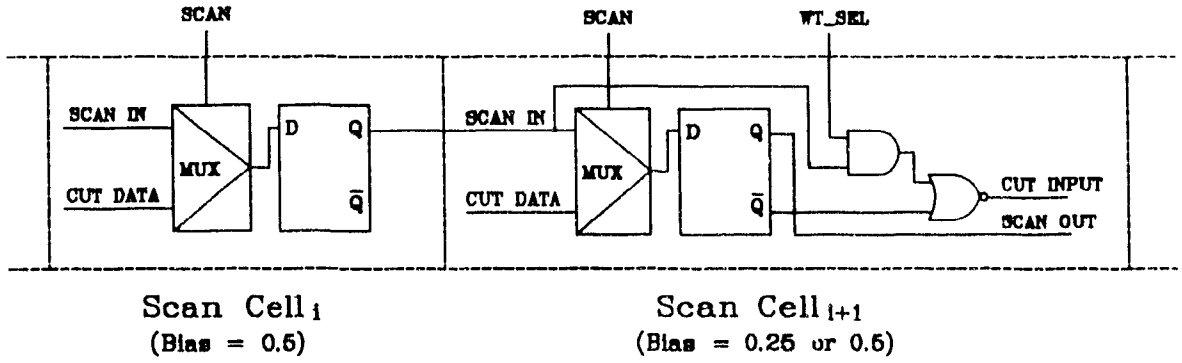


Figure 6.20 Scan Chain Ordering for Possible Missing State

Using the GSCAN cells designed, if scan cell i+1 conditionally generates a bias of 0.25, during weighted mode the pattern '11' can never occur at positions 'i,i+1'. Similarly, if cell i+1 conditionally generates a bias of 0.75, during weighted mode the pattern '10' can never occur at positions 'i,i+1'. Note that the state which is absent depends on the particular logic design used to produce the weighted output of the GSCAN cell.

A possible remedy to ungeneratable states is as follows. By observing the test vectors for the difficult faults missed, the sites where the missing input state is critical can be identified. If the corresponding modified scan cell is replaced by a generic one (i.e. the

bias is relaxed to 0.5) the problem is eliminated with little effect to the test length if the number of positions relaxed is small. Further investigation into the impact of correlation introduced by modifying the scan chain and due to LFSR generation is required

6.6 CONCLUSIONS

A circuit independent weighted random pattern generation scheme was proposed in this thesis. Here, a uniform random sequence and a single weighted random sequence was shown to be highly effective in testing very large circuits containing up to thirty eight thousand lines. The off-line weighting process is based on existing fast fault simulation techniques, and can be easily incorporated into most existing design automation environments (development of additional sophisticated algorithms is not required)

The testing scheme is geared to be suitable for a BIST application By modifying specific scan cells, the BIST hardware conditionally generates the weighted bit stream *locally* at specific input sites This design concept can also be adapted to a global test generation strategy and external testing

Apart from demonstrating that in the cases examined, one weight set was sufficient for a notable decrease in test length, it was also noticed that a very coarse weight set (i.e. restricting biases to 0, 0.25, 0.5, 0.75 and 1) provides excellent results Using finer resolution within the weight set usually results in a shorter test length but at the expense of a much higher area penalty.

In the future, a joint scheme involving test point insertion may provide the means of attaining 100% fault coverage in an even shorter test length. Also, it would be intersting to examine the extent in which the correlation introduced by the BIST generation hardware and due to LFSR generation affects the target fault coverage of such large circuits.

Appendix A. Extended Experimental Results

This section contains a more detailed tabulation of results for the quantization results using all four thresholds of section 6.3.5.1 For each circuit, three tables are presented The first is the test length and coverage before and after phase 2 modification of the weight set The second and third tables are the distribution of different scan cell types and a formal breakdown of the input biases. These tables give the **absolute** number of inputs which correspond to a cell type or bias.

C6941	Before Pin Relaxing			After Pin Relaxing		
CASE	L_{rp}	L_{wrp}	# Undet. Faults	L_{rp}	L_{wrp}	# Undet Faults
1	0.5K	35K	9	0.5K	40K	7
2	0.5K	40K	8	0.5K	40K	8
3	1.8K	39K	38	1.8K	39K	42
4	0.4K	39K	36	0.4K	39K	40

Table A.1 Quantization Runs for C6941

C6941	Number of Pins Biased to Specified Levels													
	Phase 1							Phase 2						
CASE	0	.13	.25	.50	.75	.87	1	0	.13	.25	.50	75	.87	1
1	34	10	19	126	24	28	6	33	10	17	131	24	26	6
2	34	13	16	126	32	20	6	33	12	15	131	30	20	6
3	34	-	17	149	41	-	6	33	-	15	154	39	-	6
4	34	-	29	129	49	-	6	33	-	27	134	47	-	6

Table A.2 Pin Bias Distribution for C6941

C9389	Before Pin Relaxing			After Pin Relaxing		
CASE	L_{rp}	L_{wrp}	# Undet Faults	L_{rp}	L_{wrp}	# Undet Faults
1	11.5K	20.5K	0	11.5K	6K	0
2	11.5K	20.5K	0	11.5K	6K	0
3	11.5K	20.5K	0	11.5K	6K	0
4	11.5K	20.5K	0	11.5K	6K	0

Table A.3 Quantization Runs for C9389

C9389	Number of Pins Biased to Specified Levels													
	Phase 1							Phase 2						
CASE	0	.13	.25	.50	.75	.87	1	0	.13	.25	.50	.75	87	1
1	227	-	40	318	31	19	90	227	-	40	318	31	19	90
2	227	4	36	318	45	5	90	227	4	36	318	45	5	90
3	227	-	37	328	42	-	91	227	-	33	333	42	-	90
4	227	-	45	314	48	-	91	227	-	40	320	48	-	90

Table A.4 Pin Bias Distribution for C9389

C11657	Before Pin Relaxing			After Pin Relaxing		
CASE	L_{rp}	L_{wrp}	# Undet. Faults	L_{rp}	L_{wrp}	# Undet. Faults
1	5.3K	27K	9	5.3K	37.5K	4
2	5.3K	27K	8	5.3K	37.5K	1
3	5.3K	37.5K	18	5.3K	37.5K	20
4	5.3K	28K	7	5.3K	36K	8

Table A.5 Quantization Runs for C11657

C11657	Number of Pins Biased to Specified Levels													
	Phase 1							Phase 2						
CASE	0	.13	.25	.50	.75	.87	1	0	13	.25	.50	.75	.87	1
1	100	36	106	286	17	36	34	100	36	99	296	17	33	34
2	100	56	86	286	27	26	34	100	55	80	296	25	25	34
3	100	-	106	335	40	-	34	100	-	102	342	37	-	34
4	100	-	142	286	53	-	34	100	-	135	296	50	-	34

Table A.6 Pin Bias Distribution for C11657

C30989	Before Pin Relaxing			After Pin Relaxing		
CASE	L_{rp}	L_{wrp}	# Undet. Faults	L_{rp}	L_{wrp}	# Undet. Faults
1	13.5K	38.5K	56	13.5K	39K	54
2	13.5K	38.5K	58	13 5K	39K	55
3	13K	39.5K	108	12K	39.5K	121
4	11.5K	39K	73	11.5K	39 5K	73

Table A.7 Quantization Runs for C30989

C30989	Number of Pins Biased to Specified Levels													
	Phase 1							Phase 2						
CASE	0	.13	.25	.50	.75	.87	1	0	.13	.25	.50	.75	.87	1
1	261	20	114	1117	75	47	34	261	20	108	1134	70	41	34
2	261	39	95	1117	95	27	34	261	38	90	1134	85	26	31
3	261	-	85	1202	86	-	34	261	-	77	1218	78	-	34
4	261	-	134	1124	115	-	34	261	-	128	1137	108	-	34

Table A.8 Pin Bias Distribution for C30989

C35002	Before Pin Relaxing			After Pin Relaxing		
CASE	L_{rp}	L_{wrp}	# Undet. Faults	L_{rp}	L_{wrp}	# Undet. Faults
1	15K	38.4K	17	15K	38.4K	17
2	15K	30.8K	25	15K	30.8K	25
3	15K	30.8K	30	15K	30.8K	30
4	15K	30.8K	25	15K	30.8K	25

Table A.9 Quantization Runs for C35002

C35002	Number of Pins Biased to Specified Levels													
	Phase 1							Phase 2						
CASE	0	.13	.25	.50	.75	87	1	0	.13	.25	.50	.75	.87	1
1	422	11	41	834	19	5	132	422	11	40	837	19	5	130
2	422	19	33	834	21	3	132	422	19	32	837	21	3	130
3	422	-	51	844	15	-	132	422	-	50	847	15	-	130
4	422	-	52	835	23	-	132	422	-	51	838	23	-	130

Table A.10 Pin Bias Distribution for C35002

The results of simulating the modeled mixed-weighted scheme with varying biases on the WT-SEL control line, are presented in table A.11. The coverages at 50K and 70K are given.

NETLIST	W_SEL$_{bias}$	#Undet$_{50K}$	#Undet$_{70K}$
C6941	75%	19	15
	87	27	17
	94	26	19
	97	32	29
C9389	6	5	1
	9	4	1
	13	4	1
	25	10	7
	50	13	10
C11657	63	23	14
	70	25	15
	75	23	15
	85	31	23
C30989	50	101	57
	63	95	55
	70	101	62
	75	95	71
C35002	25	62	48
	37	58	28
	50	71	36
	63	68	41
	70	92	51

Table A.11 Extended Results for the modeled BIST Implementation with 32 bit LFSR

Figures A.1 through A.5 compare the test application time for the mixed-weighted simulation and the uniform random pattern testing. Application time and test length are linearly related, however these figures are given since the log scales used when discussing test lengths may tend to obscure the magnitude of the actual time saved. A test frequency of 2MHz is used.

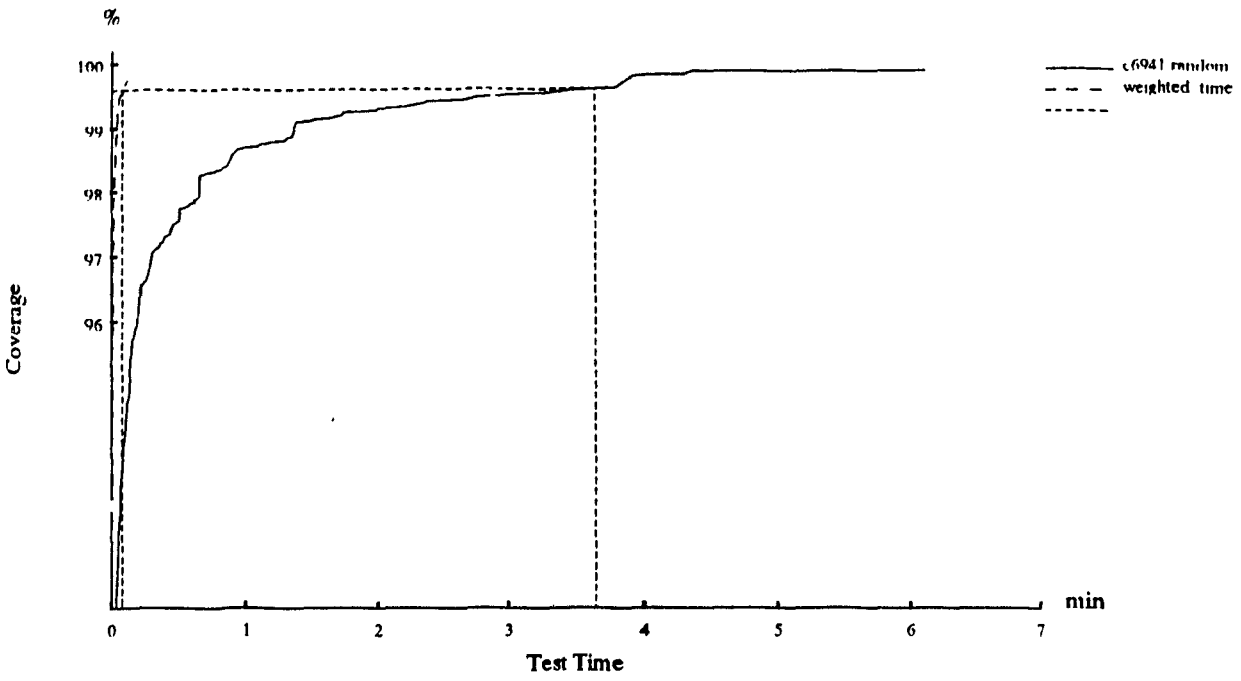


Figure A.1 Coverage vs. Application Time Plots for C6941

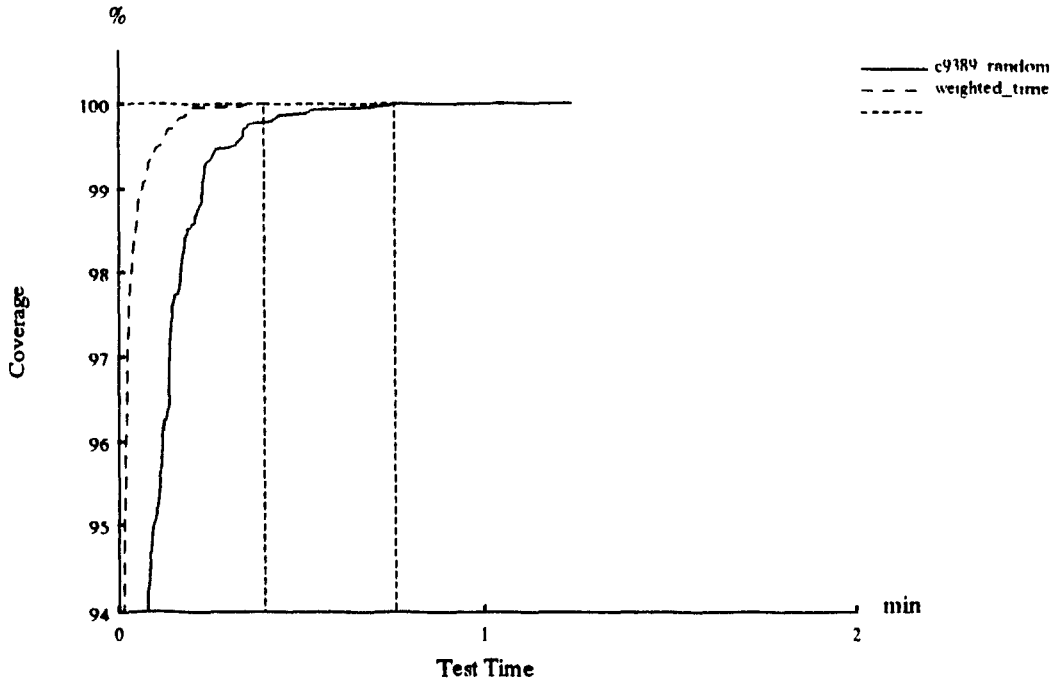


Figure A.2 Coverage vs. Application Time Plots for C9389

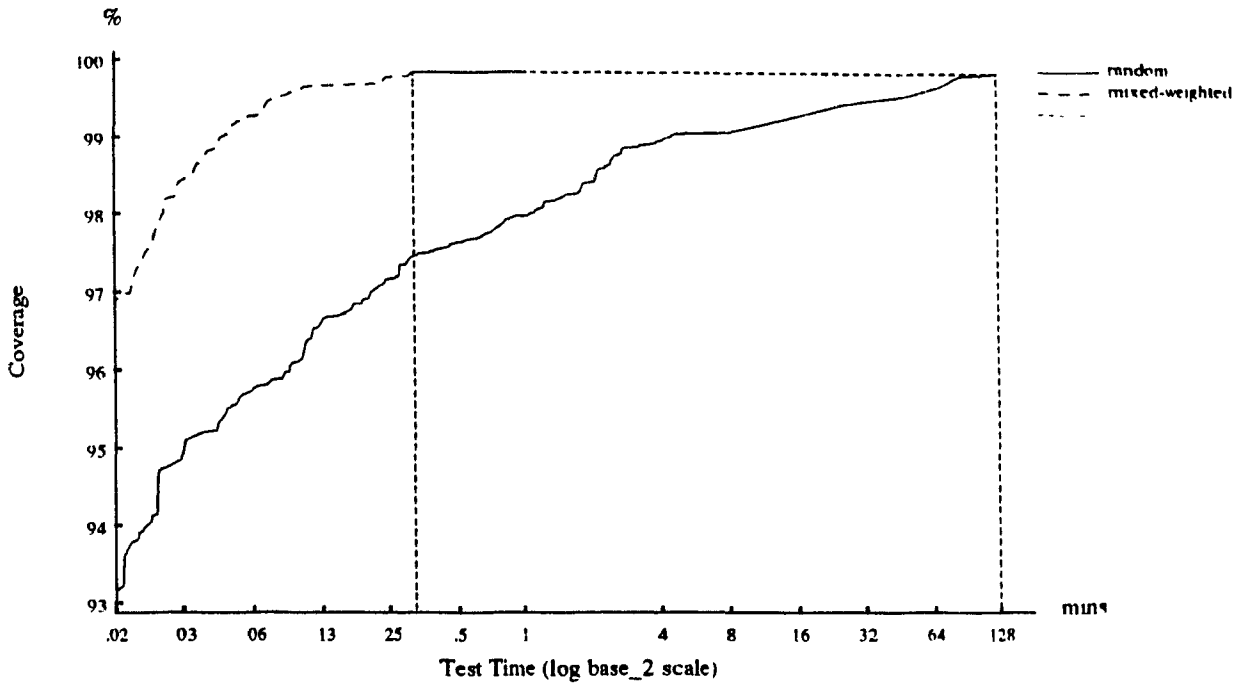


Figure A.3 Coverage vs Application Time Plots for C11657

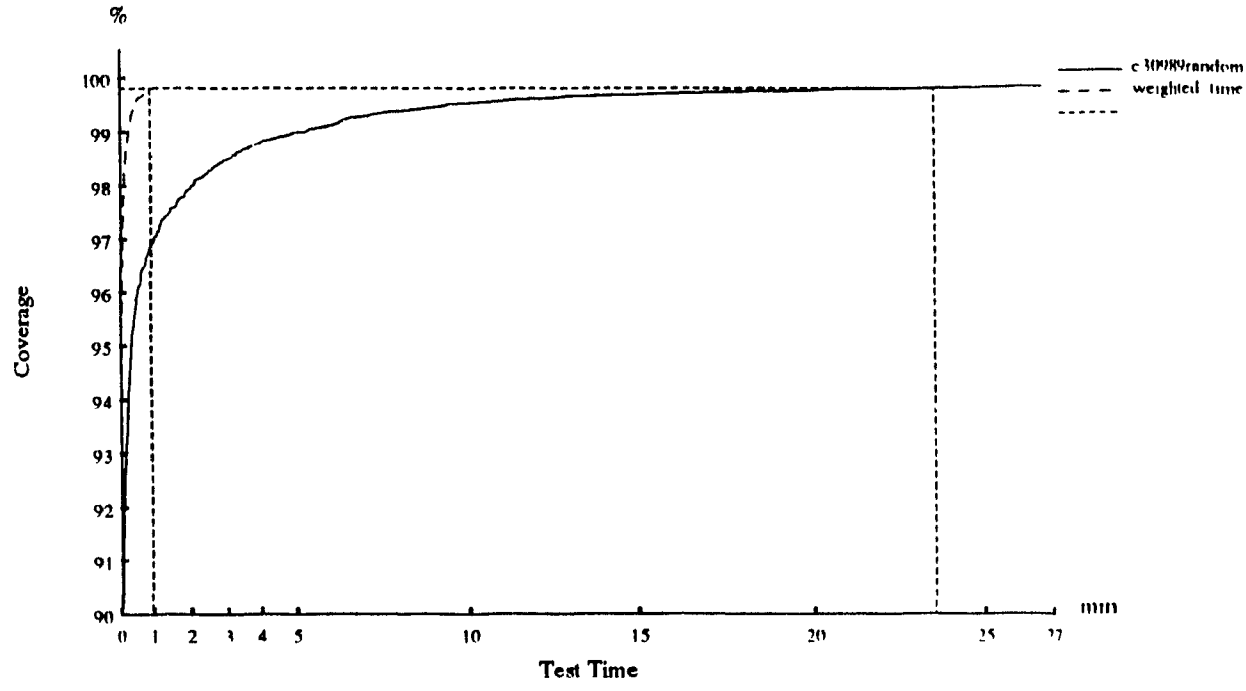


Figure A.4 Coverage vs. Application Time Plots for C30989

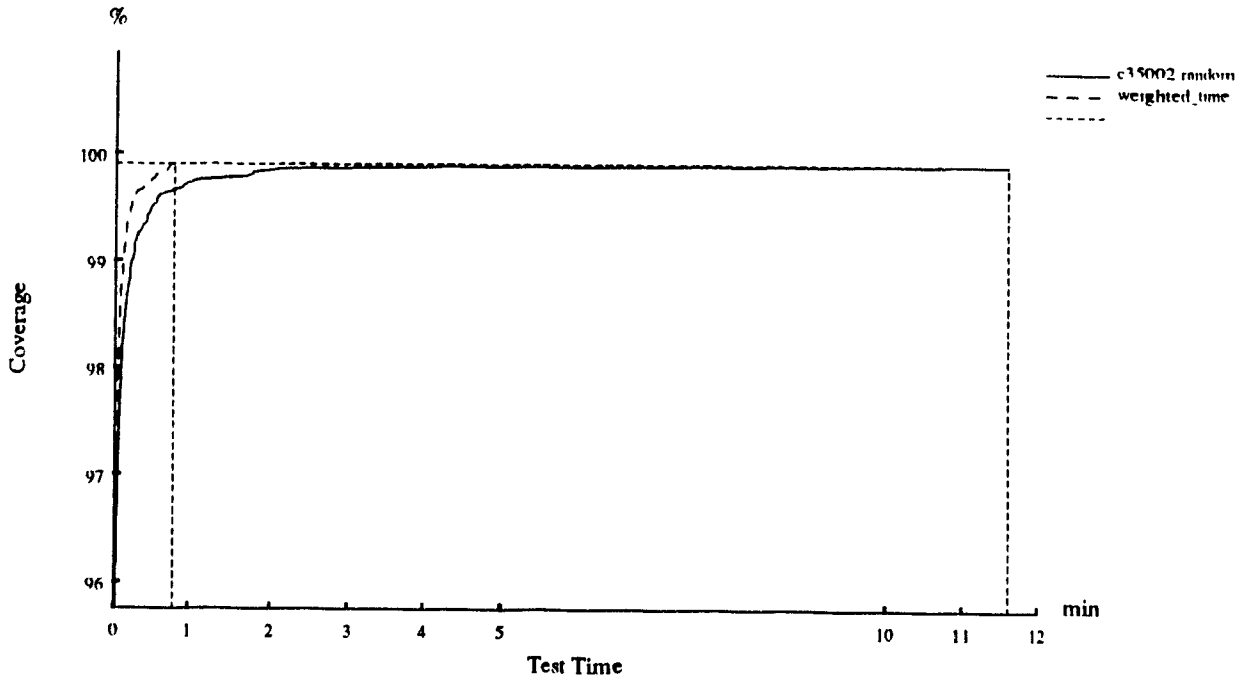


Figure A.5 Coverage vs Application Time Plots for C35002

References

[Abo83] M.E. Aboulhamid and E Cerny, "A Class of Test Generators for Built-In Testing," *IEEE Trans. Computers* Vol. C-32, No. 10, pp 957-959, Oct 1983

[Aga81] V.K. Agarwal and E. Cerny, "Store and Generate Built-In Self-Testing Approach," *Proc FTCS-11* pp 35-40, 1981

[Agr82] V.D. Agrawal and M.R. Mercer, " Testability Measures - What Do They Tell Us?," *Proc 1982 IEEE Test Conf*, pp. 391-396, 1982

[Ait90] R. Aitken, *PhD Thesis*, McGill University – in preparation

[Ake76] S. Akers, "A Logic System for Fault Test Generation" *IEEE Trans on Computers*, vol c-25, NO 6, pp 620-630, June 1976.

[And80] H. Ando, "Testing VLSI with Random Access Scan," *Proc COMPCON Spring 1980* pp. 50-52, 1980.

[Bar87] P.H. Bardell, W.H. McAnney, J. Savir, *Built-In Self-Test for VLSI.* Wiley-Interscience, New York, 1987.

[Bas89] R.W Bassett et al., "Low Cost Testing of High Density Logic Components," *Proc. ITC 89*, pp 550-557, 1989.

[Bha89] D. Bhattacharya, B.T. Murry, J.P. Hayes, "High-Level Test Generation for VLSI," *IEEE Computer.* Vol 22, No 4, pp. 16-24, April. 1989

[Boz80] S. Bozorgui-Nesbat and E. J. McCluskey, "Structured Design for Testability to Eliminate Tet Pattern Generation," *Proc FTCS-10.* pp. 158-163, 1980

[Bra84] D. Brahme and J.A. Abraham, "Functional Testing of Microprocessors", *IEEE Trans Comp* , Vol C-33, pp 475-485, June 1984.

[Brg84] F. Brglez, "On Testability Of Cominational Networks," *Proc. IEEE Symp Circiuts Sys* , pp 221-225, May 1984.

[Brg89] F. Brglez, C. Gloster, G. Kedem, "Hardware-Based Weighted Random Pattern Generation for Boundary Scan," *Proc ITC-89*, pp 264-274, 1989

[Car82] W.C. Carter, "The Ubiquitous Parity Bit," *Proc. FTCS-12*, pp 289-296, 1982.

[Che89] K T. Cheng and V.D Agrawal, "An Economical Scan Design for Sequential Logic Test Generation," *Proc FTCS-19*, pp 28-35, 1989

[Chi87] C.K Chin and E.J McCluskey "Test Length for Pseudorandom Testing," *IEEE Trans Computers.* Vol. C-20, pp 1286-1294, Nov. 1971.

[Cox90] H. Cox and J. Rajski, "A Method to Calculate Necessary Assignments in Algorithmic

Test Pattern Generation" - to appear, *Proc ITC-90.*

[Dav80] R. David, "Testing by feedback Shift Register," *IEEE Trans. Conputers*, Vol. C-29, pp. 668-673, July 1980

[Eic87] E.B. Eichelberger et al., "Weighted Random Pattern Testing Apparatus and Method," United States Patent # 4687988, Aug 18, 1987

[Fed86] X. Fedi and R. David, "Some Experimental Results From Random Testing of Microprocessors," *IEEE Trans Inst & Meas*, Vol IM-35, No 1, pp 78-86 March 1986

[Fro77] R.A. Frowerk, "Signature Analysis: A New Digital Field Service Method," *Hewlett Packard Journal*, pp 2-8, May 1977

[Fun89] S.Funatsu, M Kawai, A Yamada, "Scan Design at NEC," *IEEE Design and Test*, Vol 6 No. 3, pp 50-57, June 1989.

[Fuj83] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms," *IEEE Trans Computers*, Vol. C-32, pp1137-1144, Dec. 1983.

[Glo88] C.S Glostei and F Brglez, " Boundary Scan with Cellular-Based Built-In Self-Test," *Proc ITC-88*, pp 138-145, Washington DC, Sept 1988.

[Gra89] D Graham, *Introduction to Multi-Strategy Testing*, GenRad Inc., Concord MA, 1989.

[Goe81] P Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Trans Computers*, Vol. C-30, pp 215-222, Mar 1981

[Gol79] L H Goldstein, " Controllability/Obeservability Analysis of Digital Circuits," *IEEE Trans Circuits and Systems*, Vol. CAS-26, pp. 685-693, Sept 1979

[Gol67] S.W Golomb, *Shift Register Sqeuences*, Holden-Day Inc., San Francisco CA, 1967

[Har87] D. Harel and B. Krishnamurthy, "Is There Hope for Linear Time Fault Simulation?," *Proc FTCS-17*, pp 28-33, July 1987

[Has88] A Hassan, J Rajski and V K Agarwal, "Testing and Diagnosis of Interconnects using Boundary Scan Architecture," *Proc. ITC-88*, pp. 126-137, Washington D C, Sept. 1988.

[Hor89] P D Hortensius, R D. McLeod, "Cellular Automata-Based Signature Analysis for Built-In Self-Test" *Third Technical Workshop - New Directions for IC Testing*, pp. 117-128, Halifax, Oct. 1988

[Hor89] P.D Hortensius, R.D. McLeod, W Pries, D.M Miller, and H.C. Card, "Cellular Automata-Based Pseudorandom Number Generators for Built-In Self-Test," *IEEE Trans Comp Aided Design*, Vol. 8, pp. 842-859, Aug. 1989

[Hon81] S-J Hong and D L Ostapko "A Simple Procedure to Generate Optimum Test Patterns for Parity Logic Networks" *IEEE Trans Computers*, Vol. C-30, pp. 356-358.

[How89] M.C Howells, R.C Aitken, and V.K. Agarwal, "Defect Tolerant Interconnects for

VLSI," in *Defect and Fault Tolerance in VLSI Systems*, I. Koren, ed., pp. 65-76, Plenum Press, New York, 1989.

[Hur88] S.L. Hurst, "A Hardware Consideration of CALBO Testing," *Third Technical Workshop – New Directions for IC Testing*, pp 129-146, Halifax Oct 1988

[ISC85] ISCAS85 Benchmarks, Special Session, "Recent Algorithms for Gate-Level ATPG with fault Simulation and their Performance Assessment," *Proc IEEE Int Symp Circuits Syst*, June 1985.

[Iva88] A. Ivanov and V.K. Agarwal, "An Iterative Technique for Calculating Aliasing Probability of Linear Feedback Shift Registers," *Proc FTCS-18*, Tokyo, Japan, June 1988

[Iye89] V S Iyengar and D Brand, "Synthesis of Pseudo-Random Pattern Testable Designs ," *Proc. ITC-89*, pp. pp. 501-508, Washington DC, Aug. 1989.

[Kar85] M Karpovsky, ed. *Spectral Techniquesand fault Detection*, Academic Press, London

[Kim88] K. Kim, D.S. Ha, J.G. Tront, "On Using Signature Registers as Pseudorandom Pattern Generators in Built-in Self-Test," *IEEE Trans Comp Aided Design* Vol 7, pp 919-928, Aug 1988.

[Koe79] B Koenemann, J Mucha and G Zwiehoff, "Built-in Logic Block Observation Techniques," *Proc ITC-79*, pp. 37-41, Cherry Hill NJ., 1979

[Kra87] A. Krasniewski and S Pilarski, "Circular Self-Test Path: A Low-Cost BIST Technique," *Proc 24th Design Automation Conf.*, pp 407-415, 1987

[Kri85] B. Krishnamurthy and C. Sheng, "A New Approach to the Use of Testability Analysis in Test Generation, *Proc ITC-85*, pp. 769-778, 1985

[Kri84] B. Krishnamurthy and S.B. Akers, "On the Complexity of Estimating the Size of a Test Set," *IEEE Trans Computers*, Vol. C-33, No. 8, 1984

[Kub83] J.R. Kuban and W.C. Bruce, "The MC6804P2 Built-In Self-Test," *Proc ITC-83*, pp 295-300, 1983

[Kaw89] M. Kawamura, K. Mera, A. Tateishi, " A Study on Efficient Test Generation for Large Scan-based Circuits," 12th IEEE Workshop on Design for Testability, Vail, CO, April 1989.

[Lis87] R. Lisanke, F. Brglez, A. Degeus and D. Gregory, "Testability-Driven Random Test Pattern Generation," *IEEE Trans. CAD*, Vol. CAD-6, No.6, Nov 1987.

[Maa88] F Maamari and J Rajski, " A Fault Simulation Based on Stem Regions ," *Proc ICCAD*, pp. 170-173, Nov 1988.

[Man89] W.R. Mann, "R96MFX Test Strategy," *Proc ITC-89*, pp 611-614, Washington D C , Aug. 1989.

[McC84] E.J McCluskey, "Verification Testing – A Pseodo-Exhaustive Test Technique," *IEEE*

Trans Computers, Vol C-33, pp. 541-546, June 1984.

[McC85a] E.J. McCluskey, "Bulit-In Self-Test Techniques," *IEEE Design and Test*, Vol. 2, No. 2, pp 21-28, April 1985

[McC85b] E J. McCluskey, "Bulit-In Self-Test Structures," *IEEE Design and Test*, Vol. 2, No. 2, pp 29-36, April 1985

[Mei71] K C Y Mei, "Bridging and Stuck-at Faults," *IEEE Trans Computers*, Vol. C-23, pp. 720-727, July 1974

[Nad88] B Nadeau-Dostie, A. Ermarkaryan, L McNaughton, "Practical Scan Design for ASIC's," *Third Technical Workshop – New Directions for IC Testing*, pp 169-184, Halifax, Oct 1988

[Nil80] N.J Nilsson, "Principles of Artificial Intelligence," Morgan Kaufman Pub , Calif., 1980.

[Par89] K P Parker, "The Impact of Boundary Scan on Board Test," *IEEE Design and Test*, pp 18-30, Aug 1989.

[Pet72] W W Peterson and E.J. Weldon, *Error Correcting Codes*, 2nd ed , MIT Press, Cambridge MA, 1972

[Pra88] M M Pradhan, E.J O'Brien, S.L. Lam and J Beausang, "Circular BIST with Partial Scan" *Proc ITC-88*, pp 719-729, Washington DC, Sept. 1988.

[Rot66] J P Roth, "Diagnosis of Automata Failures: A Calculus Method," *IBM J Res Devel*, Vol 10, pp. 278-281, July 1966.

[Sav84] J Savir, G S Ditlow and P Bardell, "Random Pattern Testability," *IEEE Trans Computers*, Vol C-33, pp. 79-90, Jan. 1984.

[Sch75] H D Schnurmann, E Lindbloom, R.G Carpenter, "The Weighted Random Test Pattern Generator," *IEEE Trans Computers*, Vol. C-24, pp. 695-700, July 1975.

[Sch88] M H Schulz and E Auth, "Advanced Automatic Test Pattern Generation and Redundancy Identification," *Proc FTCS-18*, Tokyo, Japan, June 1988.

[Sel68] F F. Sellers, M.Y Hsiao and L.W Bearnson, "Analysing Errors with Boolean Difference," *IEEE Trans. Computers*, Vol. C-17, pp. 676-683, July 1968

[Ser88] M. Serra, D. Miller, J C. Muzio, "Linear Cellular Automata and LFSRs Are Isomorphic," *Third Technical Workshop – New Directions for IC Testing*, pp 213-223, Halifax, Oct 1988

[Set85] S C Seth and V D Agrawal, "Cutting Chip-Testing Costs," *IEEE Spectrum*, pp. 38-45, Apr. 1985.

[She85] Shen J P , Maly W , Ferguson J F , "Inductive Fault Analysis of MOS Integrated Circuits," *IEEE Design and Test*, pp. 13-26, December 1985.

[Sia88] F. Siavoshi, "WTGPA: A Novel Weighted Test-Pattern Generation approach for VLSI Built-I Self-Test", *Proc ITC-88*, pp. 256-262, Washington DC, Sept 1988

[Smi79] J.E. Smith, "Detection of Faults in Programmable Logic Arrays", *IEEE Trans Comp*, Vol. C 28, pp 845-852, Nov. 1979

[Ste77] J H Stewart, "Future Testing Of Large LSI Circuit Cards," *Proc 1977 Semiconductor Test Symp*, pp 6-15, Oct 1977

[Tot88] K. Totton and S. Shaw, "Self-Test. The Solution to the VLSI Test Problem?," *IEE Proceedings*, Vol. 135, Pt. E, No. 4, pp. 190-195, July 1988.

[Ude88] J.G. Udell, "Reconfigurable Hardware for Pseudo-Exhaustive Test," *Proc ITC-88*, pp 522-530, Washington DC, Sept. 1988.

[Wad78] R L. Wadsack, "Fault Modelling and Logic Simulation of CMOS and MOS Integrated Circuits," *Bell Sys Tech Jour*, Vol 57, pp 1449-1474, 1978

[Wai85] J.A. Waicukauski, E.B Eichelberger, D O Forlenza, E lindbloom and T McCarthy, "A Statistical Calculation of fault Detection Probabilities by Fast Fault Simulation," *Proc ITC-85*, Philadelphia, P A, 1985

[Wai88] J.A. Waicukauski, V.P. Gupta and S T Patel, "Fault detection Effectiveness of Weighted Random Patterns," *Proc ITC-88*, pp 245-255, Washington DC, Sept 1988

[Wan86] L.T. Wang and E.J McCluskey, "A Hybrid Design of Maximum-Length Sequence Generators," *Proc ITC-86*, pp 25-37, Sept 1986

[Wil83] T.W. Williams and K.P. Parker, "Design for Testability - A Survey," *Proc. IEEE*, Vol. 71, pp 98-112, Jan 1983.

[Wil86] T.W. Williams, W. Daehn, M. Gruetzner, C.W. Starke, "Comparison of Aliasing Errors for Primative and Non-Primative Polynimials," *Proc ITC 1986*, Washington D C, 1986

[Wol83] S. Wolfram, "Statistical Mechanics of Cellular Automata", *Rev of Modern Phys*, Vol 55, pp 601-644, 1983.

[Wun87] H-J. Wunderlich, "Self-Test using Unequiprobable Random Patterns," *Proc Fault Tolerant Computing Symposium*, pp. 258-263, July 1987

[Wun88] H.-J Wunderlich, "Multiple Distributions for Biased Random Test Patterns," *Proc ITC-88*, pp. 236-244, Washington DC, Sept. 1988.