A Self-Driven Test Methodology for Built-In Self-Test of Sequential Circuits

Fidel Muradali

M.Eng., McGill University, 1990
B.Eng., McGill University, 1987

Department of Electrical Engineering
McGill University

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

December 1996

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to take this opportunity to thank my supervisor Dr. Janusz Rajski for accepting me to the Ph.D. program and for his guidance during my term of study, and Dr. Jerzy Tyszer for his advice concerning this topic and all the other personal and technical discussions. I would also like to thank Dr. Frank Ferrie for agreeing to be a member of my Ph.D. committee.

I am especially grateful to Chinsong Sul, Don Davis and Victor Zia for taking on the undesirable task of proof reading the text, and to Eric Masson for his assistance in writing the French version of the abstract. I will truly miss the camaraderie.

Finally I would like to thank my parents, brother and sister for their patience and support during my many years of study.

Abstract

Test cost comprises a substantial portion of producing an integrated circuit. As a result, structural modifications of the circuit via design for test (DFT) techniques are commonly used as an aid to reduce test cost to the lowest possible level. One important class of DFT is Built-In Self-Test (BIST). In BIST, test generation and response analysis logic is integrated into the original circuit and are transparent during normal operation. In this manner, in-circuit tests can be performed with minimal need of external test equipment, if any.

Test strategies based on pseudorandom test stimuli are attractive since the simplicity of the pattern generation logic facilitates on-chip test application. Unfortunately, until now, these methods have been more appropriate for testing combinational rather than sequential circuits. This is largely because, unlike combinational testing, detection of sequential faults can require specific orderings of circuit operations which are prohibitively difficult to produce using a pseudorandom source.

This thesis introduces a new DFT technique which permits at-speed on-chip sequential testing using parallel pseudorandom test patterns applied only to the primary inputs of the circuit under test. Test network design focuses on adjusting fault free circuit activity and aiding error propagation. This is done via the strategic insertion of a small number of low area test points. The resulting system is unique in that aside from a test mode flag, all I/O signals needed for test system operation are tapped from within the circuit itself. This feature virtually eliminates the control signal generation logic typically needed in other test point strategies. Also, as opposed to the conventional approach of restricting circuit alterations to the state elements, the proposed flexibility in choosing modification sites is beneficial when considering speed constrained designs.

Experiments demonstrate that high single stuck-at fault coverage is achieved for a number of benchmark circuits.

Résumé

La vérification d'un circuit integré résulte en une partie importante de son coût total de fabrication. En conséquence, des techniques de modifications structurelles sont souvent appliquées aux circuits pour en simplifier leur vérification et ainsi en réduire les coûts. L'auto-vérification est l'une de ces techniques qui est fréquemment utilisée en pratique. Elle consiste a générer un test et en faire son analyse à partir du circuit lui-même sans en gêner son cycle normal d'opération. Cette méthode réduit au minimum le besoin d'appareils de test externes qui sont souvent très dispendieux.

Les méthodes de tests qui dépendent de vecteurs pseudo-aléatoires offrent aussi des résultats intéressants grâce à leurs simplicité de conception sur circuit intégré. Cependant, les résultats jusqu'à présent n'étaient satisfaisants que pour les circuits sans automates séquentiels. Ceci est une conséquence directe de la grande difficulté à générer de façon pseudo-aléatoire une séquence de vecteurs nécessaire à la détection d'un défaut séquentiel.

Cette thèse présente une nouvelle technique de modifications structurelles qui rend pratique la vérification de circuits séquentiels à partir de vecteurs pseudo-aléatoires placés aux entrées externes du circuit sous inspection. Et ce, à la vitesse normale d'opération du circuit. La synthèse du réseau de vérification a pour objectif d'ajuster l'activité du circuit sans défaut tout en améliorant la propagation des erreurs. Ces réseaux sont obtenus en insérant un faible nombre de points de vérification comportant une petite fraction de la superficie du circuit original. Tous les signaux requis pour la vérification sont générés à partir du circuit lui même mis à part un signal d'activation de vérification qui est de source externe. Ceci rend cette méthodologie unique en son genre et la distingue des autres techniques d'insertion de points de vérification car elle élimine le besoin de multiples signaux de contrôle et les circuits qui leurs sont associés. En outre, la méthode ne se limite pas à la modification de la représentation binaire des élements séquentiels et permet donc ainsi des choix plus optimaux par rapport à la performance temporelle du circuit.

L'efficacité des idées proposées a été vérifiée a l'aide de circuits étalons.

Claim of Originality

This thesis presents new ideas, techniques and results in the area of digital testing as follows:

- A new control design concept in which rather than deriving control signals from reference sources external to the functional logic, appropriate signals are tapped from lines already existing within the circuit under test.

- In the self-driven control idea described above, adjustment of fault free switching and error propagation enhancement are used to define a new DFT methodology and implementation which enables at-speed pseudorandom testing of sequential circuits. As a marked departure from existing schemes for pseudorandom sequential testing, the technique does not specifically aim to provide access to state elements. Instead, as the method is based on test point insertion, any circuit line is a candidate for modification.

- In order to curb computational effort, much of the required analysis is conducted in the fault free domain. The initial problem identified is the need to establish a threshold level of circuit switching and line sensitivity. The sole role of controllability modifications is to achieve this goal. Thus, all of the analysis associated with controllability point insertion is performed in the fault free domain. This is unlike many existing techniques which consider controllability points to aid fault activation and error propagation. The new implementation, however, does not preclude the use of this latter approach of controllability point insertion. Observability points transfer an error signal to another internal circuit line. By choosing the latter line as one not correlated to the faults from which the error signal can be mapped, fault free observability measures are used to rank the effect of the candidate connections. Of course, the procedure can degrade to the use of fault domain observability but this is at higher computational expense. The implementation of observability points is also novel in that, as part of the self-driven test network, each point is designed and chosen such

that the switching previously enabled by self-driven controllability points remains relatively unaltered.

- As a whole, the new interactive algorithm proposed can be considered dedicated to constructing the self-driven system, or parts can be used to address problems common to testing, such as test point placement, determination of circuit nodes which are pairwise independent and calculation of sequential observabilities. A new application of fault-free path tracing is instrumental to these algorithms. In addition, it was necessary to develop fast and simple testability estimates which provide sufficient accuracy to rank the effect of tentative circuit modifications. To do so, the effect of a test point is modeled as a shift in signal probability at a potential insertion site. Testability estimation then focuses on propagating this probability offset through the circuit. As reconvergence generally exists, correction factors based on pre-sampled data and intermediate sensitivity information are suggested.

- The new test system offers flexibility in that 1) the particular known initial state is not important, 2) compared to conventional test point approaches, there is a significantly broader range of signal probabilities which can be used for control purposes, 3) there are usually a number of alternative choices for modification sites, 4) rather than using the suggested procedures, existing test point insertion schemes can be used to provide an intermediate test point solution which is then converted into a self-driven final implementation.

Details of this work has been presented at the 14th IEEE VLSI Test Symposium, 1996 [Mur96].

Chapter 1 Introduction

An integral part of producing a microelectronic device, or any system for that matter, is assuring that the complex fabricated design works. The field of testing responds to this concern.

In modern circuits, addressing test issues can be the most time consuming part of the design task and can comprise in excess of a third of production costs [Bha89]. In fact, device testing expense can increase by an order of magnitude (about a factor of ten) per level of packaging, eventually to the point where thousands of dollars can be at stake for field tests of installed systems [Bar87]. Thus, in order to curb long term test costs to the lowest possible level, component tests should be as thorough as possible from the earliest stages of assembly (e.g. wafer and chip-levels). In addition, rather than treating the test problem purely as an independent post-fabrication issue, given a predetermined test methodology, the design of the circuit itself should be geared to facilitate decreased effort and increased test effectiveness. Such linking of design and test processes via analysis and manipulation of a structural circuit representation is termed design for testability[1] . An example of this is built-in self-test (BIST) wherein on-chip and/or on-board circuits provide and analyse test data.

Given current circuit densities, DFT is invaluable in terms of reducing test effort. Compared to a circuit designed for functionality, it is generally desirable that DFT accounts for only minor contributions, if any, to parameters, such as circuit size, operational speed degradation and power consumption. Also, the DFT modification technique should be fast and easy to

---

[1] The same concept applied at higher levels of description is commonly known as *synthesis* for testability.

implement - preferably via automation as this largely insulates the user from the complexity of the test re-design process and related structures. Furthermore, since designs and standards can vary over time, it is beneficial if the underlying procedures are easy to understand or at least flexible enough to allow inclusion into other hybrid DFT schemes and adaptability to unforeseen custom circuits. For instance, a competitive circuit might require qualities including high speed performance which can in turn impose stringent design constraints on the number, size and location of additional test logic. In such scenarios, along with satisfying the above objectives, the feasibility of a proposed DFT technique can be facilitated if it maximizes the freedom in selecting possible modification sites.

Simply, testing is done by applying a set of stimuli to input pins, and examining the circuit outputs for consistency with the expected result. The effort of the procedure can be coarsely divided into three parts: determination of the input set, the time to apply the test and evaluation of the test results. This thesis addresses the first two issues. The goal is to provide a low area overhead DFT strategy which permits rapid testing of a fabricated sequential circuit. Furthermore the test should be suitable to a BIST environment.

The proposed approach endeavors to address the problems encountered when testing a sequential circuit with parallel pseudorandomly generated test patterns applied only to the circuit under test's (CUT's) primary input pins. Recognizing that the sequential test problem is particularly difficult because of the large input space to be searched for a solution, many existing pseudorandom-testing DFT schemes utilize state access methods which temporarily transform the sequential test problem into a less computationally intensive combinational one. Even when sequential functionality is retained, commonly, modifications remain limited to providing some form of state access. In doing so, sometimes area penalties can be large and the possible reformatting of input data can result in high test times.

The new approach succeeds, in part, by recognizing that with pseudorandom stimulation, a major test hurdle is that the number of traversable states is constricted. A number of strategically inserted test points remedies this situation. The test network implementation is novel in that, apart from a test mode flag, it is not controlled by signals derived from sources

external to the existing functional logic. Instead, suitable controls are extracted from within the CUT itself. Circuit modification is not restricted to the state elements and the test is conducted in full sequential mode at the normal operating speed of the circuit. Results show a marked improvement over pseudorandomly testing an unmodified circuit and DFT techniques previously used to enable similar test execution formats.

The structure of the rest of this thesis is as follows: Chapter 2 offers definitions and a brief background concerning test generation, test application and response compaction issues required for the discussion. Chapter 3 outlines some of the concepts of testability and reviews some alternative DFT structures which have been adapted to pseudorandom testing of sequential circuits. Chapter 4 introduces the new DFT approach along with pertinent experimental results. Finally, Chapter 5 summarizes some of the qualities of the new test technique, and formulae required for circuit analysis are presented in the Appendix.

Chapter 2 Background & Test Issues

A digital circuit is one which responds to a discrete set of voltages applied at its input lines, and, in turn, asserts another set of these *digital signals* at its output lines. In general, the value of a digital signal is restricted to logical 1 or logical 0, corresponding to within a threshold of the power supply and ground potentials respectively. A single enumeration of input signals is called an *input vector* or *input pattern*. Similarly, a corresponding collection of output line values is an *output vector* or *output pattern*. Digital circuits are classified as either combinational or sequential in operation. A combinational circuit contains no memory, thus an output response depends only on the input vector applied. On the other hand, because of the existence of memory, successive output vectors of a sequential circuit are related. This implies that a specific sequence of input vectors may be needed to force a sequential circuit to a particular output state. Likewise, consecutive states of a sequential circuit are interrelated.[1] This interdependence among circuit states can cause difficulty in automatically generating the input patterns required for testing sequential circuits.

2.1 Failures & Fault Models

Testing is the process by which a manufactured structure is checked for correctness. This is done by applying a set of input *test patterns* to the circuit and comparing the observed result

[1] In general terms, a state refers to a subset of line values occuring at a particular time. Unless specified as otherwise, this thesis adheres to typical assumption that these values are sampled at the individual memory elements (latches or flip flops).

with an expected response. A mismatch implies that the circuit contains either physical defects or design errors which render it unacceptable. The complete set of test patterns is termed the *test set* and the number of patterns which comprise the test set is the *test length* .

This thesis addresses the problem of detecting the presence of physical defects. Testing for design errors, such as incorrect mapping between different levels of abstraction and inconsistencies between the device design and design specifications, is termed *design verification testing* and is not considered for elaboration.

Defects are physical anomalies which can cause a circuit to malfunction. These may be a result of imperfections in the fabrication or assembly processes 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, electron migration and environmental conditions, such as radiation, vibration, humidity and temperature.

It is generally not feasible and unnecessary to explicitly enumerate and analyse each possible defect scenario. As such, fault models are devised to be behaviorally representative of many electrically significant defects [She85]. Although a single model which can characterise the entire universe of such failures has not yet been formulated, considering the detection of faults rather than defects succeeds in reducing the amount of analysed cases to a comparatively manageable number while achieving a high quality test.

Based on their stability in time, faults may be classified as permanent, intermittent (present only at certain times) and transient (occurs only once). Faults can also be classified as *static* or *dynamic* depending on the manner in which the circuit is affected. Static faults, also known as *logic faults*, cause malfunctions which affect the steady state logical operation of the circuit. A fault of this type is usually detected independently of circuit delays by applying a single test vector and allowing a sufficient time for the circuit switching to settle before sampling the associated output response. Thus, the rate at which test vectors are applied may be slower than the normal operating speed of the CUT. An example of a static fault model, which is currently

the most commonly used, is the *single stuck-at* fault [Eld56]. Here, under the influence of a fault, a line is held at a fixed value irrespective of the polarity of the signal driving it. Hence, the line is said to be 'stuck' at either logical 1 (s-a-1) or logical 0 (s-a-0). As in most other fault models, a second stipulation is that only a single fault can exist is the circuit. This is done to curb computational effort when devising a test set since the number of possible faulty cases to be analysed is limited to $2n$, where n is the number of circuit lines. Alternatively, if multiple faults are assumed, the number of faulty representations grows exponentially in the order of 3^n-1 since a line may be s-a-1, s-a-0 or fault-free. Crosspoint faults in programmable logic arrays [Smi79] and bridging faults representative of a pair of shorted lines[Mei74] are examples of other static faults not explicitly handled by the stuck-at model.

Dynamic faults, or *parameteric faults*, may not change static operation of the circuit but affect a circuit parameter causing a violation of a design specification concerning, for example, operational speed, current levels or voltage levels. Detection of a dynamic fault can require time-dependent monitoring of output responses or electrical parameters. Test vector ordering may also be necessary. Such faults include gate and path delay faults [Par90] which cause the circuit to operate at a rate slower than anticipated. Another example is IDDQ faults[Sod89] which cause a higher than expected leakage current. Stuck open faults [Wad78] result in abnormally high resistance (ideally non-conducting) transistors or connections. During testing, these faults can be considered dynamic in nature since the delay of the circuit can be increased and because leakage can cause the circuit to reach the correct value given enough time (e.g. a stuck open pull down). Note however, some stuck open faults exhibit static behaviour, for instance an inverter with a stuck open pull-up transistor will permanently hold an output of 0 once that value is reached. Stuck short faults indicate permanently conducting transistors. As with stuck open faults, while these faults can affect logical circuit behaviour, dynamic treatment is more complete.

The stuck-at model was originally suggested for vacuum tube circuits. While these products are now obsolete, the model remains applicable to modern integrated circuits. Many authors though have questioned its continued use and propose alternative fault models such as those cited above. Nevertheless, the single stuck at model remains popular because of its

computational viability, availability of analysis techniques and technology independence. In addition practical experience indicates that, especially when tests are applied at the operational speed of the circuit, a complete stuck-at test can also detect a large number of other fault types[Wil83] [Gre94]. Moreover, in cases where testing is done according to a set of fault models, stuck-at is usually one of them [Max92]. For these reasons, this thesis considers the detection of single stuck-at faults.

Once a fault model is decided upon, the effectiveness of a test set is measured as the proportion of total faults detected. This value is termed the *fault coverage* of the test. However, it should be noted that fault coverage is a lower bound on the probability that all defects are covered. Another measure which relates to the quality of tests products is the *defect level* [Wil81] - the percentage of circuits which are incorrectly judged as not defective. This measure is more difficult to obtain since it requires information concerning process yield. The indication though, is that thorough testing according to many fault models is significant in reducing the defect level.

2.2 Approaches To Testing

There are two approaches to generating test vectors depending on whether a functional or structural circuit representation is processed [Gra89][Man89].

Functional testing verifies that the circuit operates according to specification, therefore, test effectiveness relies on in-depth familiarity with circuit behaviour. The problem can be addressed in two ways. In the first, neglecting explicit hardware details, possible defects and fault models, the circuit and its internal modules are hierarchically checked for expected functionality and interaction. For example, at the logic-level, adders must combine bits properly, memories can be accessed and ALUs should perform all desired operations. In the alternate approach, tests are derived according to functional fault models. Model definition should be realistic in the sense that the faulty behaviour induced is similar to that caused by defects or accepted low-level fault models (i.e. to the extent that the detection of the latter is

covered by tests generated for the functional fault set.) Such translation is vital since test quality measures are usually expressed in terms of low-level faults. An intermediate approach assumes that almost any fault can occur. Thus, the circuit is exhaustively tested by completely exercising the fault-free behaviour (e.g. considering combinational circuits, application of 2^n patterns, where n is the number of input pins). Unless partitioning is employed, this option is only feasible for small circuits. This concept of exhaustive and pseudoexhaustive testing will be revisited in Section 2.6.2. Functional tests are commonly used for design verification.

The premise of *structural testing* is that a circuit will operate correctly if each of its components (gates, interconnects, etc.,) is fault-free. Test development requires knowledge of the operation of only the basic circuit elements (e.g. logic gates, lines and simple blocks, such as flip flops). Fault models are employed to assign possible faults to the components, and thus the procedure can be automated.

2.3 Test Pattern Generation & Fault Detection

A *controlling* input value to a gate forces a specific gate output value irrespective of all other input assignments to that gate. Given a *non-controlling* input value to a gate, the other gate inputs determine the value at the output. For instance, an input 0 to an AND gate is controlling while an input 1 is non-controlling. A gate input is said to be *sensitive* if all other inputs to that gate are non-controlling. In effect, inverting the value at a sensitive gate input changes the value at the gate output. The ability to assign values, especially non-controlling values, to internal lines is integral to the test process. As will be seen later, the difficulty in accomplishing this from primary inputs sometimes results in the need for design modification.

2.3.1 Test Vector Operation

Testing a circuit node involves a *path sensitization* process in which the input test data accomplishes the following tasks:

- stimulates the CUT inputs to force, or control, the desired line to the known fault-free value, and
- stimulates the CUT inputs so that the effects of this activation can be propagated and observed at an output node.

Figure 2.1 illustrates the principle.

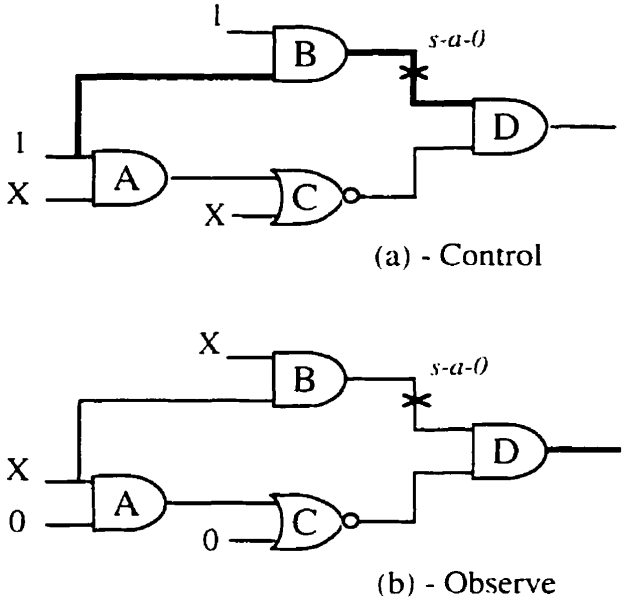


Figure 2.1: Operation of a Test Vector -
(a) Control the fault site; (b) Observe the effect.

In Figure 2.1a, a s-a-0 (stuck at 0) on the output of gate B is to be tested. The input assignment 11xx controls this line to a fault-free value of 1. Next, in order to observe the effects of the potential fault, a path must be sensitized from the faulted line to the output. A circuit path with all side inputs set to non-controlling values comprises one type of sensitized path. Thus, since the path to the PO is along the output of gate D , the output of gate C must be set non-controlling. This is done in Figure 2.1b by the input vector xx00. Since there is no conflict between the vectors used to control and observe the fault site, the final test pattern is 1100. This example is a simplified version of the test generation procedure since, for example, multiple paths may be required for error propagation. The mechanics of test pattern generation is also omitted. Procedural details can be found in [Abr90].

Generating a test vector is an NP-complete problem [Iba75]. This implies an inherent worst case complexity which is exponential in the size of program input - in this case the number of circuit inputs. Fortunately, on average, practical test generation tools run in polynomial time. Existence of redundant faults (those for which no test pattern exists) and signal reconvergence typically cause these algorithms to exhibit worst case behavior.

Complete information for testing a combinational fault can be contained within a single test pattern. However, for sequential circuits, a specific sequencing of circuit operations, thus a plurality of test patterns, may be needed to achieve the internal line values required to control and/or observe a single fault site. Furthermore, the initializability and the operation of memory, often through complex feedback loops, can make the effect of a fault difficult to determine. Specifically, the complexity of sequential test generation grows exponentially with the length of circuit cycles and linearly in sequential depth [Che89] [Gup90] [Wun89a]. As a result, sequential tests are typically longer and tests are considerably more difficult to generate than those used for combinational testing.

2.4 Automated Test Pattern Generation (ATPG)

Automated approaches for generating test patterns rely on a description of the CUT constructed in software for easy manipulation. In *fault-oriented* ATPG, appropriate modeled faults are associated with each node. The objective of *fault independent* ATPG is to set up conditions within the circuit which are suspected to detect faults without explicitly targeting them. Related to the manner in which test patterns are selected/constructed, there are two approaches: *deterministic test pattern generation* (DTPG) and *random-pattern-based test generation* (RPTG).

2.4.1 Deterministic Test Pattern Generation (DTPG)

Deterministic test pattern generators analytically devise a test vector by implementing a path sensitization process. Most heuristics are based on branch and bound techniques used to search

through the input space while curbing the amount of computational overhead. In this process, the number of operations required to undo incorrect decisions tends to dominate the computational cost.

2.4.1.1 Combinational DTPG

Methods for combinational DTPG have existed for roughly three decades with some of the earliest work contained in the D-algorithm [Rot66]. Improvements and alternatives to this include the use of algebraic analysis via the boolean difference, multi-valued logic for increased decision resolution and identification of necessary assignments, restricting the manner in which implications are made and preprocessing to identify pivotal node assignments. The use of at least one of these techniques can be found in methods, such as those of [Sel68] [Ake76] [Mut76] [Goe81] [Fuj83] [Sch88] [Raj90] . Usually, combinational ATPG techniques are the foundation of sequential DTPG systems.

2.4.1.2 Sequential DTPG

Most sequential ATPG approaches consider synchronous circuits. That is, circuits in which state updates occur only when specified by the system clock, thus operations are *synchronized* to the clock. Algorithms can be categorized in terms of the level of abstraction used to describe the circuit. Commonly, these representations are at the gate-level, state transition-level, and register-transfer-level (RTL).

2.4.1.2.1 Gate-Level Sequential DTPG

At the gate-level, the most common strategy is a transformation of the time sequential behaviour into a related space sequential case. As such, the use of combinational techniques is enabled. This concept of *time frame expansion* is demonstrated in Figure 2.2.

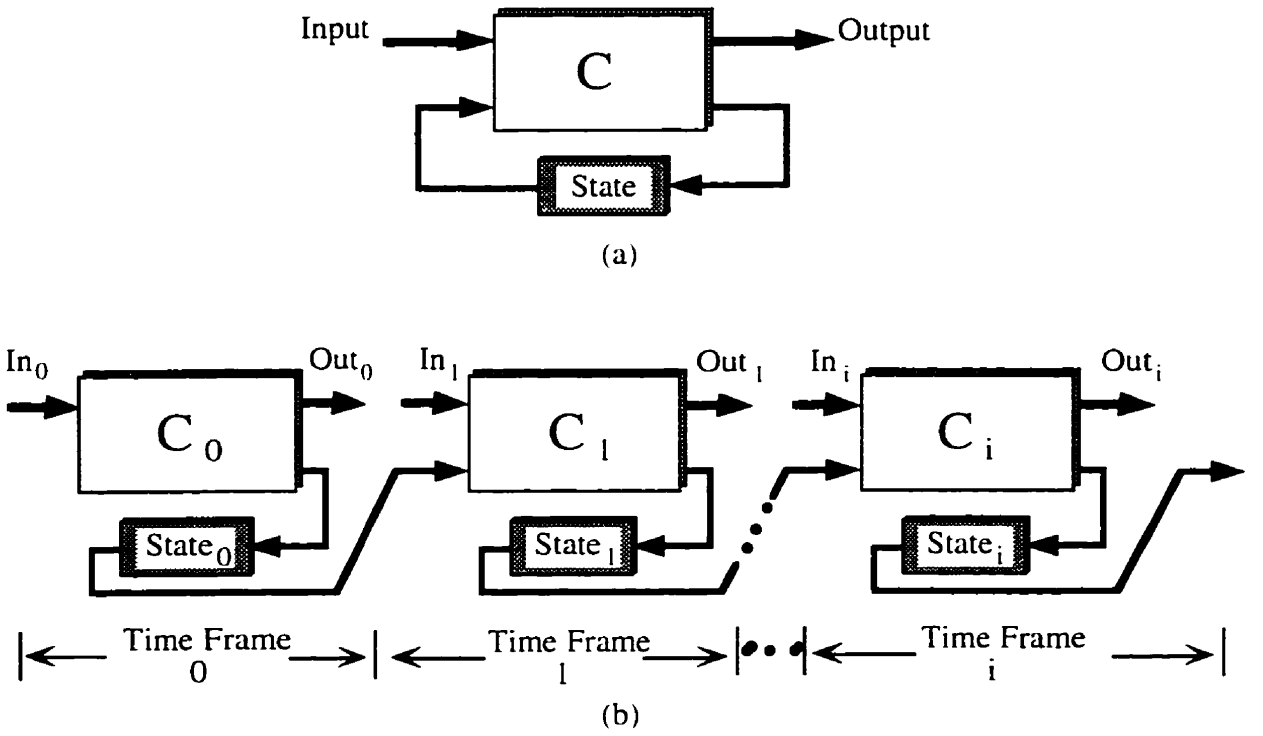


Figure 2.2: Time Frame Expansion - (a) standard Mealy model of a sequential circuit; (b) time frame-expanded combinational model.

In Figure 2.2b, a combinational model of the sequential circuit is formed by creating copies of the circuit logic, the inputs to which are the PI's, state element outputs and regenerated feedback signals from previous copies of the circuit. This manner of expanding, or "unfolding", the circuit over a number of time frames, where each time frame corresponds to a clock pulse, effectively removes time sequential behaviour. Combinational DTPG algorithms can now be used to generate tests for this new structure, however, an original single stuck-at fault is also duplicated in each frame of the iterative network. Therefore, an applicable combinational DTPG routine is one which is capable of handling large circuits containing multiple faults [Mut76]. Also, because feedback is broken and regenerated, it is possible that test vectors generated according to this model will cause races in the actual circuit [Che92]. Similar test invalidation problems can arise when asynchronous circuits are processed. Test vector verification via fault simulation is one remedy to this problem.

Various time frame expansion techniques have been developed differing in the "direction" in which the sequential circuit is unfolded and stored in memory. *Forward time processing* is the most straightforward approach and determines test vectors in the order in which they are applied. There, assuming a known initial state, an appropriate number of time frames is preselected and combinational test pattern generation for multiple faults is applied. The upper bound on the required number of time frames is 4^n , where n is the number of state elements.[2] However, as memory usage grows linearly with circuit size and the number of combinational copies, the practical time frame limit is much smaller than the maximum bound.

In the more general case, because the initial circuit state is not known a priori, *mixed time processing* can be used to generate *self-initializing* sequences [Put71]. In this case, it is not necessary to predetermine the number of time frames (although a practical limit can be imposed). An initial copy of the circuit is created at time 0 and a combinational test is attempted in a forward processing manner. As needed, previous time copies (negative time copies) are created to justify initial values at the flip flops instrumental to the forward phase. Since justification is in a backward or *reverse time processing* direction, the circuit is unfolded in a "mixed" manner. Without revision, memory usage can be worse than in pure forward time processing.

A solution to high storage costs is offered by algorithms which rely predominantly on reverse time processing[Mar78] [Mal85] [Che89]. In the earliest method [Mar78], for a given fault site, a potential propagation path to a PO is selected. The anticipated propagation may require a number of time frames. Starting from the PO, the selected path is sensitized backwards. If successful, justification of the value required at the fault site proceeds. Upon failure, another propagation path is chosen and the process is restarted (of course, partial information can be saved for reuse). Although the number of paths attempted per fault can be high, the gain is that, at any time, only two time frames are needed - the current and previous. Thus only two copies of the circuit are resident in memory.

---

[2]Given n flip flops, for each of the 2^n states in the good machine, the faulty one may be in one of its 2^n states. Thus, the maximum length of a test sequence containing no repeated states is no greater than 4^n[Abr90]. Alternatively, a maximum of 2^n state transitions are required to justify (reach from the initial state) the affected faulty/fault free states and maximum 2^n state transitions to differentiate between the faulty/fault free state pair.

A number of heuristics have been proposed to improve the space requirements of the basic approaches described above. For instance, in the RTP method of [Mal85], rather than specifying a complete path, only a subpath (as small as a single element) from the fault site to the PO is specified. Alternatively, since it is recognised that a fault may simultaneously require multiple propagation paths, only a destination PO can be chosen [Che88]. As generation effort can be wasted if many alternate POs need to be attempted, [Nie91] adopts a mixed time process which eliminates the need for selecting destination POs and complete propagation paths. Memory complexity is managed by retaining only a subset of frames needed for FTP.

Several programs [Ma88][Kel89][Sch89][Ono91][Aut92] use at least one of the following techniques to accelerate gate-level test generation: preprocessing to determine useful implications, concurrent storage and reuse of partial results, extraction of partial state transition information, the use of testability guides to estimate forward processing depth and, instead of re-initializing for each fault, performing opportunistic selection of target faults which are easily detectable from a given current state (usually the circuit state existing due to the application of previously found test vectors.)

2.4.1.2.2 Functional-Level Sequential DTPG

When dealing with large circuits, detailed analysis of a gate-level structure can be avoided by performing test generation using a higher-level circuit description. Another advantage of this approach is that tests created at higher levels are independent of hardware fault models yet applicable to any implementation of the machine. This permits test set development to potentially commence before a final logic-level description is available. Test generation using a state transition graph (STG) circuit description is such an alternative.

Usually, as in checking experiments[Hen64], high-level approaches would be categorized as functional tests, however, the distinction is somewhat lost in newer methods which integrate both gate-level and higher-level ATPG. Moreover, in order to attain high gate-level coverage and exploit STG test generation procedures, hardware faults are sometimes mapped to functional faults in the high-level description.

2.4.1.2.2.1 Checking Experiments

One of the earlier sequential testing approaches is based on a CUT's STG specification. Given a reference STG containing no equivalent states (*reduced*) and such that there exists an appropriate path along which any state is reachable from another (*strongly connected*), the assumption is that occurring faults alter the STG without increasing the number of distinct states. A checking sequence is an input sequence that distinguishes a given *n*-state machine from all other machines with the same inputs and outputs and at most *n* states. Existence of checking sequences is guaranteed by the following theorem [Moo56] [Abr90]:

> For any reduced strongly connected n-state sequential machine M, there exists an input-output sequence that can be generated by M, but cannot be generated by any other machine M' with n or fewer states.

Thus, the test sequence is a functional test which essentially traverses the STG and checks for the existence of all states and the correctness of the state transitions. If M can be initialized to a single known state via a *synchronizing sequence*, a checking sequence can be derived by determining a *distinguishing sequence* - i.e. a single sequence which produces different output responses for each initial state[Hen64]. Unless a general STG is modified, the existence of a distinguishing sequence is not assured and, if it does exist, the upper bound on its length is exponential in the number of states.

An improvement is based on Unique Input/Output (UIO) sequences[Sab88]. These are significantly shorter[3] than distinguishing sequences and the definition is less general - a UIO_i for state S_i is one which differentiates S_i from all other states in the STG. Verification is similar in that upon traversal it can be determined that the STG is in state S_i by applying and monitoring the response to UIO_i.

The checking problem can also be cast into that of determining the equivalence between two STGs. For example, considering completely specified circuits with known initial states, the faulty and fault-free STGs can be concatenated and state minimized[DeM94]. If the reduced STG has the same number of states as the original, the machines are equivalent.

---

[3]This is justified experimentally - no UIO upper bound has been formally proven.

If the machines are incompletely specified, such equivalence checking cannot be used. In such cases, given a single output circuit, one solution is to create a faulty-fault-free product STG by XORing the outputs of the two machines[Gho92b]. Thus, a test sequence is an input sequence which causes the product STG to traverse from the initial state to a state which asserts an output 1. Fault models used to corrupt the STG are discussed in the following sub-section.

2.4.1.2.2.2 Non-Checking STG Test Generation

Compared to checking experiments, a more systematic approach is to derive tests for faults injected into a STG. Faulty STGs can be extracted from faulted logic-level descriptions using explicit or implicit enumeration techniques [Gho92b][DeM94]. Alternatively, functional STG-level faults can be injected to corrupt the transition arcs of a good STG [Che90] [Pom91]. Given n states and m transitions, the number of such single transition arc faults is $m(n-1)$ and the number of multiple arc transition faults is (n^m-1) [Che90].

It has been found that single transition arc faults relate well to single stuck-at coverage[4] and many transistor-level faults[Che90]. However, since the number of single transition faults is exponential in the number of flip flops, fault collapsing techniques are proposed in [Che90] [Pom91]. As demonstrated in the experiments of [Pom91], for 100% coverage, it may be required to also inject a small number of multiple transition arc faults and finally , after gate-level fault simulation, extract faulty STGs for yet undetected stuck-at faults. It is noted from this final phase that, contrary to the previously accepted notion, tests for single transition arc faults may not cover detection of high multiplicity transition arc faults.

In [Pom94], stuck-at faults from 2-level implementations (actually Boolean covers) of the combinational parts of the circuit are mapped to faults in the STG. As the 2-level *pseudo-implementation* used for fault extraction is only of analytic interest and is not necessarily the final implementation, concurrence between test development and design implementation can persist. However, for the same reason, 100% stuck-at coverage is not guaranteed unless the same 2-level implementation is used directly or is transformed to multi-level logic for the final

[4]Experiments in [Che90] report 97%-100% stuck-at fault coverage for tests targetting single transition fauts.

implementation. As a stand-alone method, it is found that by considering only mapped stuck-at faults, the size of the fault space is less than that enumerated by collapsed single transition fault experiments. Despite this, the overall extraction effort can become computationally prohibitive so integration into a procedure which first injects transition faults is preferred.

Since the state table represents functional transfer of information from flip flop to flip flop, many routines conduct all or part of the test generation process at the STG-level. At the STG-level, the test tasks are:

initialization - identification of a sequence to bring the machine to a known initial state. As a simplification, this may be done using a hardware reset.

activation - identification of (1) an excitation state from which the faulted transition can be activated, and (2) an excitation vector which, from this state, activates and propagates the fault to either an output or the next state variables.

state justification - identification of an input sequence which causes traversal from the initial state to the excitation state.

state differentiation - identification of a differentiating sequence for a pair of states S_1,S_2 (S_{good}, S_{faulty}). If this sequence is applied when the circuit is initially in S_1, the last pattern of the sequence produces a different output response than if initially the state was S_2.

Because working entirely in the fault domain involves constructing and analyzing new STGs for each fault instance, a common speedup is to conduct at least part of the process (usually justification and/or differentiation) in the fault-free domain [Ma88] [Che90] [Gho91]. As a result, in at least one phase of the procedure, only the fault-free STG is needed and, concurrent with test generation or via preprocessing , useful information concerning its structure, such as partial propagation sequences, can be stored for repeated use. For instance, given a gate-level description, [Ma88] uses a PODEM-based[Geo81] time-frame expansion to find an excitation state for the fault and a propagation path (fault domain analysis). Then, using

an extracted complete or partial fault-free STG, a justification sequence is found to bring the circuit from the reset state to the excitation state.

Indeed, the use of fault-free information is an approximation, so invalid test sequences may be generated if a fault affects an assumed fault-free justification or differentiating path. A simple solution relies on fault simulation for test verification and to signal the need for alternate sequences. After fault simulation, another fix is to revoke the fault-free approximation and consider the faulted STG for appropriate state transfer sequences. STG-based methods are only applicable to circuits of limited size, such as controllers [5]. In some instances, rather than extracting a complete or partial STG, more compact representations, such as fault-free Boolean covers for each output and next state function, can be implicitly enumerated, stored and operated upon (i.e. intersected) to find consistent transfer sequences [Gho91]. However, this procedure can also become expensive in terms of processing time and memory. In addition, the fault-free assumption may result in invalid test sequences for the same reasons as outlined above. Larger circuits can be considered using RTL and gate-level analysis[Hil77] [Gho90]. It is assumed that both circuit representations are available since RTL extraction from a low-level circuit description is not yet possible. Given a gate-level fault, similar to the STG methods cited previously, a typical RTL procedure begins by generating a combinational test for one time frame. Fault free justification and propagation are then performed using the more compact RTL model.

2.4.2 Testing With Random Sequences

The premise of random-pattern-based test generation is that a useful set of test patterns are contained within a lengthy sequence of randomly selected input patterns. While the input generation procedure is relatively simple, some practical issues related to RPTG include determining the test length required for a high fault coverage and evaluating the effectiveness of the test sequence. More guided approaches consider extracting or constructing a test set from analysis of circuit activity caused by trivial manipulations of randomly generated input vectors.

[5]For instance a circuit with as few as 16 flip flops spans up to 65536 states.

2.4.2.1 Fault Insertion

Fault insertion [Sus73] is one of the earliest means of compiling and evaluating a test set. The procedure entails physically injecting a fault into a built circuit and monitoring the response to a test sequence. The advantage of this approach is the reliability of the experiment - the faulty behaviour of the circuit is accurately retrieved since an actual circuit is used. Of course, for integrated circuits, complete fault insertion is not economical or even realistic. Also, test development is not concurrent with the design process since a prototype device must be available before test tasks can commence. Node accessibility via design principles, such as the use of boundary scan (chapter 3), can enable the limited use of fault insertion for simulation tool development.

2.4.2.2 Simulation-Based TPG

Using a description of the circuit implemented in software, simulation algorithmically determines circuit behaviour. Fault simulation predicts circuit responses when faults are injected into the circuit description. Thus, mimicking the operation of a faulty/fault-free circuit, fault simulation can be used to evaluate (or *grade*)the effectiveness of generated test sequences or assist in determining the sequences themselves. However, as opposed to fault insertion, the accuracy of simulation depends on the reliability of the simulation model (e.g. the effect of delays may be omitted).

The simplest approach to fault simulation is serial fault simulation. There, for each fault, the fault-free circuit is transformed into a unique faulted version and simulation proceeds. Various advances in fault simulation include: parallel fault simulation[Ses65], where all faults are simulated in parallel for a given pattern; deductive fault simulation [Arm72] and concurrent fault simulation [Ulr74], where good and faulty values are logically deduced; parallel pattern single fault simulation [Wai85] which uses word length to encode fault data, thus reducing simulation time; differential fault simulation [Che89a] [Nie90] which avoids high memory usage by storing only the faulty values occurring at the state elements, and creating subsequent faulty machines

relative to previous faulty versions rather than the good machine; and methods, such as [Maa90] [Gou91] [Moj93], which exploit structural circuit information to accelerate simulation. Some new methods also achieve higher performance by targeting characteristics specific to particular classes of circuits. For example, [Kas96] examines the structural regularity and functionality of arithmetic and logic blocks commonly found in data-path circuits to perform very fast functional simulation on those blocks.

2.4.2.2.1 Non-Adaptive Simulation-Based Test Pattern Generation

DTPG usually targets the detection of a specific fault per iteration. Fault simulation is usually integrated into the DTPG algorithm and periodically enabled in order to identify faults incidentally detected by the already determined portion of the test set. Removal of these faults (or *fault dropping*) from a list of undetected faults can result in a significant reduction in the subsequently considered fault space. In addition, as mentioned previously, a fault simulation call is required to evaluate test sequences generated using shortcut approximations, such as fault-free initialization, justification or propagation (Section 2.4.1.2.2.2).

Deterministic applications aside, a simple simulation-based approach of random pattern test generation (RPTG) is as follows: (1) Fault simulate a number of test vectors, and (2) For each vector, if any modelled fault is detected, drop the fault from further consideration and retain the vector as a useful test pattern. The attempted vectors are generated according to some simple criteria such as pseudorandom selection [Bar87].

Frequently, there exists a group of faults which require unacceptably long test lengths for detection. These faults are said to be *pattern-resistant* with respect to the input pattern sequence. Reasons for pattern-resistance are :

- Redundancy : The fault is not detectable.
- Reconvergent fanout: Fault effects can cancel one another.
- High fan-in: It may be difficult to set values appropriate for propagation.
- Test vector quality : The generated distribution of 1 and 0 assignments is in conflict with what is required for fault detection. Also, correlation between successive input patterns or inter-bit correlation, for example due to structural or functional dependencies within the generator [Bar89], can preclude certain logic assignments.

- Initialization and STG/circuit connectivity : Poor initial state selection and the existence of loops can result in a constricted spread of traversable states.

For sequential circuits, it is common that a significant portion of faults are pattern resistant, therefore, the standard RPTG method is not usually effective for sequential testing. In contrast, combinational testing often succeeds in detecting a large percentage of faults within a relatively small number of input vectors.

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

$$P_N(X) = \prod_{f \in F} (1 - (1 - p_f(X))^N) \qquad \text{Equation (2.1)}$$

where, $P_N(X)$ is the threshold probability that each fault f is detected in the test sequence. $p_f(X)$ is the detection probability of the fault f [Gol74] [Brg84] [Set85]. This is simply a measure of the odds of detecting f. It has been found that only the faults with detection probabilities roughly within a factor of 2 of the lowest detection probability within the fault set are pivotal in the test length estimate [Sav84]. Since here $p_f(X)$ is dependent on the relative distribution, X, of 1's to 0's at each bit position of the input sequence, it may be possible to reduce the test length by manipulating X. Such *weighted random generation* [Sch75] [Lis87] [Eic87] [Sia88] [Wun88] [Mur90a-c] [Pat91] [Aga94] strategies are highly effective for combinational circuits but there has been little success for sequential structures.

A typical hybrid test generation procedure initially uses RPTG followed by DTPG to detect the pattern resistant faults. Again, this is not well suited to sequential testing due to the previously mentioned ineffectiveness of RPTG in addressing the sequential test problem . A suitable joint DTPG-RPTG test is outlined in the recent sequential testing method of [Abr92] . There, by holding each newly entered deterministic state and applying a number of pseudorandom test patterns at the PIs, a pseudorandom combinational test is embedded within the sequential test generation procedure.

2.4.2.2.2 Adaptive Simulation-Based Test Pattern Generation

An adaptive approach of simulation-based test generation constructs a test set by combining internal circuit information extracted from fault simulation and cost function guidance [Ses65] [Agr89] [Hat92] [Saa92]. The basic procedure is as follows:

- Generate and simulate a number of trial vectors
- Based on simulation results, select the best trial vector such that a cost function value eventually drops below a threshold. This cost usually reflects how close the vector or vector sequence is from achieving the current objective, such as initialization, detection of fault groups or detection of an individual "hard to detect" fault. Each of these goals can require a different cost function.
- Compose a new set of trial vectors and repeat.

The initial seed vector can be arbitrary. The set of trial vectors can be created by manipulating already known test vectors via, for example, inverting bits, merging substrings of a number of vectors and copying previously generated sequences. Often, in order to reduce the chance of timing hazards during the test procedure, the group of trial vectors is generated to be a unit hamming distance from the last vector accepted. Due to limitations of the trial vector generation algorithm or if good test vectors are selected in a greedy manner (i.e. to always maximally reduce the cost), it is conceivable that the search process reaches a local minimum from which no further reduction in cost is possible. Simple solutions to this are to change the algorithm used to generate trial vectors, accept a vector which increases the cost or change the cost function to target a different objective.

Adaptive simulation-based test generation is advantageous because delays can be modeled, asynchronous circuits can be handled and implementation is simpler than that for DTPG algorithms. One of the main drawbacks is the inability to identify undetectable faults. There may also exist faults which are prohibitively difficult to activate with the sets of trial patterns. Current cost functions typically do not offer sufficient guidance to handle these pattern resistent faults.

2.4.2.2.3 Test Set Compaction

For combinational circuits, *reverse compaction* can be used to reduce the length of a test set. The assumption is that test vectors found nearing the end of test generation tend to cover faults found earlier in the procedure since the latter have less specific test pattern requirements. By fault simulating the test set in the reverse order to which it was determined, dropping detected faults after each vector and retaining only those tests which increase fault coverage, test set size can be reduced by 25% to 50%. Unfortunately, the order-sensitive nature of sequential tests renders this technique useless for sequential testing.

2.4.2.3 Non-simulation RPTG

In order to eliminate the computational overhead of fault simulation encountered in the RPTG approach of Section 2.4.2.2, a test set need not be formally extracted but assumed to be contained within the "sufficiently long" sequence. Simulation is not performed, thus the fault coverage can only be estimated. Sometimes a statistical sampling [Agr81] [Set85] is used to approximate this value. There, a random selected sample(s) is simulated using the test sequence to be evaluated. The resulting fault coverage is used to determine the overall value.

An estimate of the required test length can be found using Equation 2.1. Also, as mentioned in Section 2.4.2.2, this approach is not favorable for sequential circuits.

2.4.2.4 Fault-Independent Approaches

Apart from checking experiments, the previous techniques are *fault-oriented* in that tests are generated for injected faults. The goal of *fault-independent* test generation is to derive tests which detect a large set of faults without explicitly targeting them. The concept of line sensitivity is important to this approach.

A critical path is one in which all lines are sensitive and which terminates at a PO [Abr84]. In a path critical with respect to a test vector V, half of the single stuck faults along the path are

detected by V [Abr90]. For example in Figure 2.3, highlighted paths are critical with respect to 1100 thus the labeled single stuck-at faults are detected.

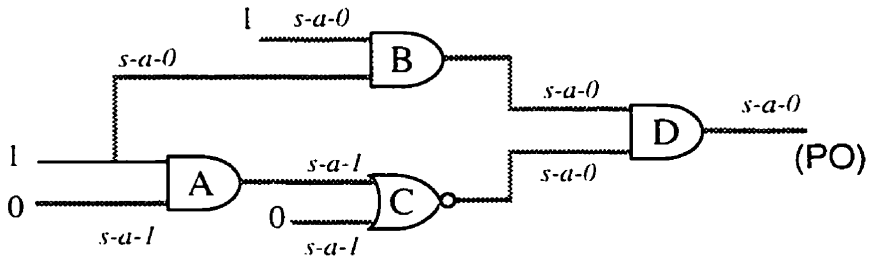


Figure 2.3: Faults Detected by Critical Paths.

Hence, it is desirable to generate tests which produce long critical paths. The basic procedure is:

1) Starting at a PO, assign it a 1 or 0 (POs are always critical)
2) Backtrace and recursively justify the PO assignment. Gates with critical outputs are justified, if possible, with critical assignments.

Critical assignments to a gate permit only one gate input to be controlling, if necessary. For example, critical input assignments to a 3-input AND are 111,011,101 and 110. Note the input assignments 0xx, 0x0 would justify an output 0 but this would render the traced path not-critical since sensitivity to a single gate input is not guaranteed.

In order to generate a set of tests, or equivalently a set of critical paths, once a critical path has been found, the recursion at step (2) returns to the last traced gate assigned critical inputs and an alternative critical combination is attempted (e.g. 011 is changed to 110).

The advantages of this approach include:

- detected faults are known without fault simulation (these can be determined from the particular critical path created)
- new tests are generated by systematically modifying existing critical paths thus much of the duplicated search effort inherent in many fault oriented algorithms is avoided.

Difficulties encountered with fault independent algorithms include [Abr90]:

- the inability to identify undetectable faults
- reconvergent fanout causes assignment conflicts, self-masking and the need for multiple path sensitization.

The suggestion that it is beneficial to favour assignment of non-controlling values in order to create sensitive paths is accepted in the new method of chapter 4. There, threshold line sensitivity is used as an objective when adjusting CUT activity.

2.5 Redundancy & Undetectable Faults

Untestable faults are those for which no test sequence exists. Such faults can be classified as combinationally untestable (or combinationally redundant) and sequentially untestable. Sequentially untestable faults can be further partitioned into *sequentially redundant* and *sequentially irredundant-but-untestable*.

Sequential and combinational redundant faults are not detectable because the terminal behaviour of the circuit remains unaffected. Inability to detect sequentially irredundant-but-untestable faults is due to the incapability of a test generation procedure to initialize the faulty or fault-free circuit. [Che91] demonstrates that faults which cause initialization difficulty are undetectable if they are not *potentially detectable*. Potentially detectable faults are detectable only from certain initial states and undetectable from others. Recently, rather than adopting the usual approach of sampling circuit outputs at a single observation time corresponding to the end of a differentiating sequence, the multiple observation time DTPG method of [Pom91b] monitors a sequence of output values. The result is the capability to detect faults independent of circuit initialization, therefore, potentially detectable faults flagged untestable by some single observation time algorithms can be covered.

There is no practical difference between an untestable fault and a testable one which is not detected by the chosen test generation process. For instance, detection of a fault may be missed because of an incomplete search process inherent to PODEM-based sequential DTPG

[Che92b] or DTPG may be terminated due excessive processing time. Similarly pattern resistant faults may remain undetected after a RPTG test.

Untestable faults waste test generation time and may preclude detection of other otherwise detectable faults[6]. It is therefore preferred to identify them before, rather during, test generation. Algorithms for redundancy identification are based on techniques, such as structural analysis [Har89], testability measures [Rai82] [Chu95], test-generation-based analysis [Thi89], symbolic simulation [Chu95], STG equivalence checking [Moo92], and identifying conflicts in necessary line assignments [Iye94] [Sch88]. Unfortunately, this problem is linked to that of test generation and is also NP-complete.

Combinational redundancies can be removed using simplification rules [Abr90] or logic minimization and resynthesis [DeM94]. Reduction of sequential redundancies can be via optimization at the STG-level [Dev89] [Dev90] [Dev91]. At the gate-level redundancy removal can be done using logic optimization during peripheral retiming [Mal91] or the sequential to combinational circuit transformation of [Che91].

Note that redundancy is not always undesirable. It can, for instance, be introduced in order to avoid hazards or aid fault tolerance. [Kra91] also shows that redundancy can be used to enhance the random pattern detectability of sub-circuits. The designer though, must still contend with the random testability of the additional logic. Test points (chapter 3) are probably a better alternative since the associated modifications can be made fully testable.Finally, in general, redundancy may not relate to undetectable stuck-at faults - it simply denotes that the circuit can be reduced. A trivial example of this is signal buffering

2.6 Test Application

Test Application effort has a notable impact on the cost of testing an integrated circuit. Most schemes can be based on either *stored pattern testing* or *hardware pattern generation. Store and*

[6]Process yield may also be reduced [Iye94].

generate and *programmed* methods are sometimes added to the list but may be treated as extensions of the two approaches listed above.

2.6.1 Stored Pattern Testing

Stored pattern testing involves the application of predetermined test vectors. While the technique is straightforward and suitable for many present circuits, especially in the case of sequential circuits, large storage capability at the tester unit can be required. In the event that buffer capacity is exceeded, large test time penalties can be incurred for buffer reload from secondary memory. This can be on the order of minutes for each reload operation [Bas89]. Furthermore, as the specifications of a tester unit are static or have restricted upgrade potential, there are physical limits imposed on variables such 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 technologies, and usually result in higher operating speeds and the need for larger test sets. The conflict ultimately implies costly tester upgrades or replacement. It is questionable then, whether using only a standard stored pattern approach is economically feasible in the long term [Bas89].

Apart from physical test application issues, the requirement of a complete or partial predetermined test set can also be a setback. This is because even with state-of-the-art DTPG tools [Nie91], for complex sequential circuits, satisfactory fault coverage is difficult to achieve within reasonable time limits [Chi90][Abr92].

2.6.2 Hardware Pattern Generation

The functional and memory requirements of the external tester can be reduced by adopting a RPTG-like strategy which utilizes test sequences that are simple to generate using relatively small uncomplicated circuits. Fault simulation is commonly performed to validate the effectiveness of these patterns.

In *pseudorandom testing*, 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 2.4.2, it is intended that after a large number of these random patterns, a sufficiently high-level of fault coverage can be achieved. Unfortunately, experience has shown that with current circuit densities, a number of test vectors orders of magnitude in excess of the maximum permitted test length may be needed to attain this goal, if it is at all possible. In fact, up until now, combinational fault detection has benefited most from pseudorandom-based techniques. The limitation in pseudorandomly testing an unmodified sequential CUT is rooted in the inability to control and observe internal lines, but can be easily understood knowing that a set of ordered vectors is needed for sequential fault detection.

The specific sequences of circuit activity required to test some sequential faults can be prohibitively difficult to reproduce using a pseudorandom pattern source. For instance, assuming that the probability of generating each member of a test sequence is 0.1, reproducing a 5 pattern sequence would require in excess of 100000 trials. Of course, there may exist a number of test sequences per fault, but the trend of effort exponentially increasing with sequence length persists[7]. Also, due to their varying influence on circuit behaviour, pseudorandomly determining certain signals, such as clocks and asynchronous controls can preclude fault detection and, in certain designs, can result in undesirable activity, such as races and oscillations.

As discussed, a modification to the basic pseudorandom test approach is the use of a joint test strategy - apply a reasonable length of pseudorandom patterns and supplement this with a set of deterministically generated stored test patterns. However, it has been found that in many combinational test cases, the size of the stored portion of the test nears 70% of a complete deterministic set [Bas89]. As sequential test sets are typically much longer than combinational ones, this approach does not properly address the problem of limited storage in either scenario. In addition, the high number of pattern-resistant faults causes sequential testing to rapidly degenerate to pure stored pattern testing.

---

[7]In effect, the example remains the same but the pattern generation probability increases slightly.

Weighted random pattern generation is especially effective in curbing combinational test lengths. However, when implemented in logic, generator hardware complexity increases with the number and resolution of input distributions. In the extreme case of incorporating the generator on-chip, it is found that for large scan-based (Section 3.2.1) combinational circuits, one coarsely biased weight can be efficiently implemented [Mur90a]. Again, weighted generation hardware is not well suited to sequential testing.

Another alternative is the use of a *store and generate* approach (e.g. [Aga81] [Abo83] [Bar85] [Fed86][Brg89][Duf91][Koe91][Ven93][Hel95][Zac95]) where, relative to the size of a deterministic set, a small number of control words are used in conjunction with a random pattern generator to produce a test sequence. Significant among these methods are recent re-seeding techniques [Koe91][Ven93][Hel95][Zac95], which are intended for scan-based combinational circuits. There, at the expense of on-board or external memory, a simple programmable finite state machine (based on an LFSR - Section 2.5.2.1) is seeded with a compact control words which, after a number of cycles, are expanded into predetermined deterministic vectors. The length and number of control words is dependent on identifying the bits per vector which are instrumental in fault detection. This can be done during DTPG or via fault simulation [Mur90a]. If the number of these needed bits is small compared to the width of the deterministic vector, the number of control words can be made less than the deterministic test length, and the word length can be orders of magnitude shorter than the deterministic vector - leading to a proportional memory saving compared to pure stored pattern testing. Again, while efficient for combinational testing, because extensive sequential DTPG analysis may be involved, store and generate approaches have not properly addressed sequential circuits.

Programmed testing is yet another extension whereby a functional test program contained in a ROM generates the required test set. This is common for microprocessor based systems [Kub83].

Exhaustive testing applies a complete functional test set. Without fault simulation, this method is guaranteed to detect all static faults, but since the test length increases exponentially with the

number of CUT inputs (2^n for combinational circuits and $(2^n)^{4^m}$ for sequential circuits[8] where n is the number of CUT inputs and m is the number of state elements), this procedure becomes impractical for circuits with as few as 25 inputs. Test lengths can made more manageable by logically or physically partitioning the CUT and testing it as a number of smaller subcircuits with reduced input widths. The practicality of such *pseudoexhaustive* approaches depends on the amount of control needed to partition the circuit sufficiently [McC84] [McC81] [Ude88][Wun89b][Wu91].

There also exists a small group of regularly structured circuits, the functionality of which enables 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.

Because of the above mentioned limitations of existing hardware generation schemes, it may seem that sequential testing defaults to a stored pattern test. However, despite these setbacks, the simplicity of using pseudorandom-based test stimuli remains attractive because of the potential for the reduction of test equipment to the point where all test hardware can be included on-chip. Chapter 4 introduces one such method.

At the root of many pseudorandom schemes, is a uniform random pattern generator. The following two subsections briefly outline two structures which can be used for this purpose.

2.6.2.1 Linear Feedback Shift Registers

A LFSR is a finite state machine comprised of a unidirectional chain of flip flops (unit delays) and XOR gates (modulo 2 adders). An example is shown in Figure 2.4.

[8] As the number of possible single input atterns is 2^n and as the test sequence lengh per fault is as much as 4^m , the exhaustive test lengh is $(2^n)^{4^m}$ - an incomprehensively large number.

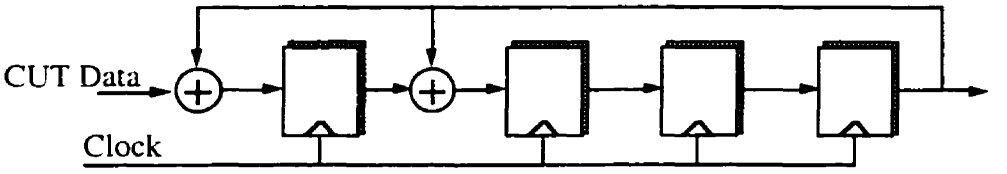


Figure 2.4: LFSR1 - polynomial divider - Characteristic Poly. $x^4 + x^3 + 1$

Such LFSR generate cyclic binary sequences by performing a linear transformation - polynomial division - on the incoming stream [Bar87]. In implementing the division, the feedback taps of an *n* -bit LFSR define the divisor or 'characteristic polynomial', and the input sequence (input polynomial) is the dividend. The *n*-bit contents of the unit after the last 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 2.5. Here, a single XOR network is positioned at the input to the first memory element of the LFSR. Both types of LFSR share the same characteristic polynomial and are isomorphic. LFSR1 is a true polynomial divider, and while both it and LFSR2 yield the same quotient stream, the remainders, or *signatures*, may differ. Also, LFSR2 is prone to slower operation depending on the size of the feedback XOR.

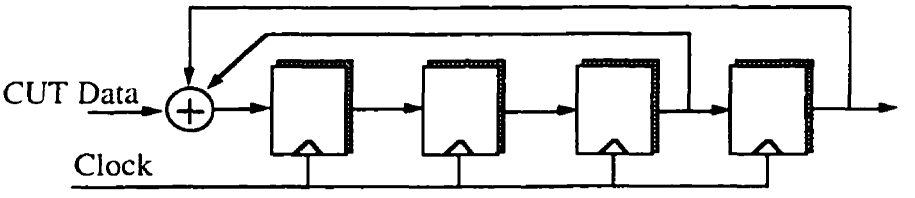


Figure 2.5: LFSR2 - Characteristic Poly. $x^4 + x^3 + 1$

An LFSR is said to be autonomous if there does not exist an external input stream to the first stage. In such configurations, the number of internal states traversed depends on the positions of the feedback taps and the initial state. If the characteristic polynomial is *primitive* , the number of unique internal states appearing in one cycle is 2^n-1 (the all zero state is excluded since it is an absorbing state). In this case the LFSR is said to be *maximal length*. As with the

construction of DeBrujin type counters, maximal length LFSRs can be configured to also include the all zero state.

One property of maximal length LFSRs is that the probability distribution at the output bits tapped at each flip flop is uniformly biased to 0.5 (the ratio 1's to 0's occurring over a maximal sequence is $\frac{2^{n-1}}{2^{n-1}-1}$). Their relatively uncomplicated structure and pseudorandom properties of maximal length sequences[Bar87] make LFSRs suitable bases for pseudorandom test pattern generators.

2.6.2.2 Cellular Automata

Another sequential structure which exhibits pseudorandom generating traits is the cellular automaton. The unit is based on 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 "rules" [Wol83] which describe the behaviour of the block. Two common examples are blocks built according to rule 90 and rule 150 (there are 256 rules). They are shown below where s[t] is the value at position i at time interval t.:

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

$$\text{Rule 150:} \quad s[t+1]_i = s[t]_{i-1} \oplus s[t]_{i+1} \oplus s[t]_i$$

A hybrid CA is constructed using more than one rule.

[Hor89][Ser88] have claimed that CA-based generators are less susceptible to the linear dependency problems sometimes encountered with LFSRs[Bar89], thus, possess superior randomness properties.

2.7 Test Response Evaluation

Conceptually, the simplest way to analyse test responses is to externally compare the entire CUT output stream with either the fault-free response found through simulation or with the corresponding outputs of a known fault-free "gold unit" [Dav76]. However, especially in the

case where the response evaluation is to be done on-board/chip, it may be desired to reduce, by orders of magnitude, the amount of data transferred to the external monitoring unit. Data compaction techniques are employed to this end.

Compaction involves operating on the output data stream using some function which either performs a transformation or extracts a qualitative feature. Currently, the most popular compaction approach is *signature analysis* [Fro77][Koe79][Dav80][Bar87], in which an LFSR performs a polynomial division-like operation on the output stream. In this transformation, all possible input bit streams are mapped onto the 2^n possible signatures of an n-bit signature analyser. Assuming that the occurrence of any output stream is equally likely and that all possible bit streams input to the compactor are mapped evenly onto the 2^n possible signatures, the number of k-bit input streams which produce the same signature is 2^{k-n}. That is:

$$\frac{2^k}{2^n} = 2^{k-n} \qquad \text{Equation (2.2)}$$

This raises an important point: since compaction is a reduction in the amount of information, there is an implied probability that useful knowledge is lost. In the above case, for every fault-free signature there are $2^{k-n} - 1$ wrong bit sequences which map to the same result. This situation in which an incorrect bit sequence yields the same signature as the good fault-free bit sequence is called *aliasing*. Aliasing can reduce test quality by causing a drop in fault coverage even though highly effective test sets are used. In addition, exact fault coverage becomes difficult to determine without extensive simulation. As the length of the compacted sequence tends to infinity (or in a practical sense becomes very large), an n-bit LFSR/CA aliases with probability 2^{-n}.

In the presence of a fault, the assumed probability of an erroneous response bit or pattern is defined by an error model. As the integrity of the uniform error assumption used above has been questioned, alternative models such as the independent error model [Wil86] and the more accurate asymmetric error model [Xav89] have been proposed. The asymptotic aliasing value of 2^{-n} is consistent with these models.

Typically an LFSR-based analyser accepts serialized circuit output data. The extension for multiple output circuits, is a multiple input signature register (MISR), an example of which is shown in Figure 2.6. The asymptotic aliasing of MISR is similar to that of LFSRs.

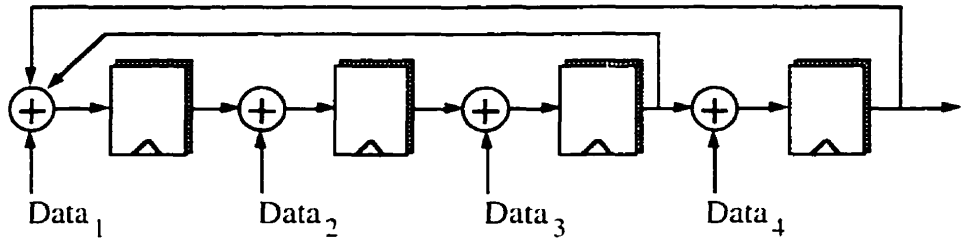


Figure 2.6: MISR - Characteristic Poly. $x^4 + x^3 + 1$

Some other transformation oriented compaction schemes are based on functional units already existing within the data path. Compared to signature analysis, the advantage of such approaches is a significant reduction in hardware overhead. For example, recent accumulator-based compaction (ABC) methods of [Raj93a] [Raj93b], demonstrate that adders can be used as low aliasing compactors.

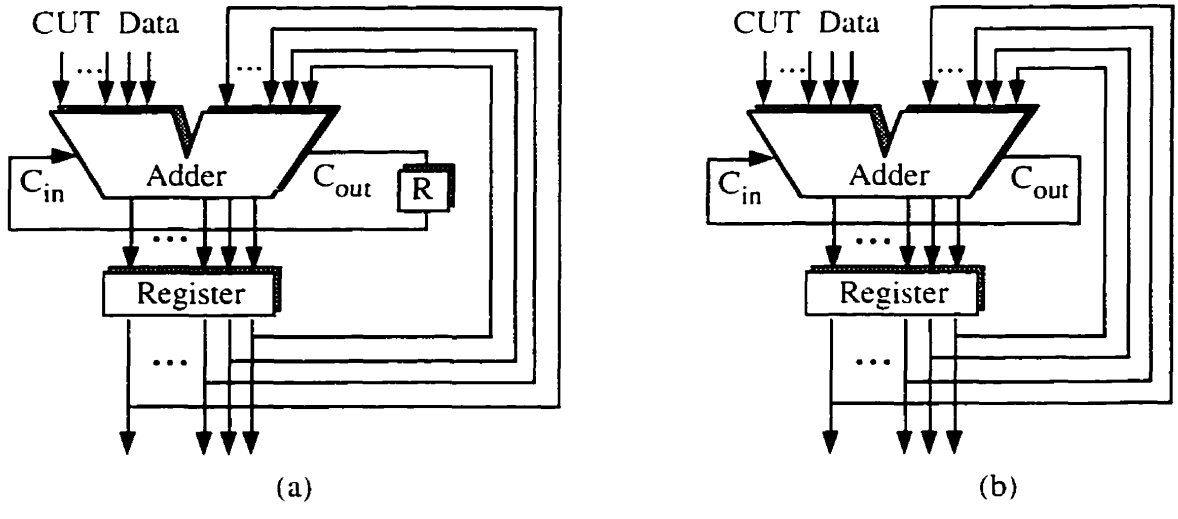


Figure 2.7: Accumulator Based Compaction -
(a) Rotate carry adder; (b) 1's Complement adder.

As shown in Figure 2.7, in test mode, adders are configured to accept CUT data at one operand field while the compactor outputs serve as the other. Information at the overflow bit is

also retained by creating a feedback connection between the carry-out and carry-in terminals. The result is an aliasing probability approximately equal to 2^{-n} for an n -bit accumulator (i.e. $\frac{1}{2^n - 1}$ for 1's complement adders and $\frac{2^n - 1 - p}{(2^n - 1)^2}$ for rotate carry adders, where p is the fault injection probability at the adder input.)

Other compaction techniques involve recognizing certain attributes of the circuit output stream, such as parity [Car82], the number of 1's occurring and its extension to exhaustive testing (syndrome testing)[Sav80], and the number of transitions occurring [Hay76]. As in [Zor90], these methods can also be combined with signature analysis.

In the previously mentioned compaction approaches, over time, a long response stream is compacted into a smaller word. Data is provided and processed per clock cycle. However, the width of the incoming data can be excessively large to interface with the compaction unit. In such cases a reduction in data bandwidth, or *space compaction*, is required. The idea of space compaction is also important when testing requires monitoring a large number of internal circuit nodes. By merging the information from these points to a single (or narrower) stream, manageable hardware overhead can be maintained.

A simple zero aliasing example of space compaction, which is still somewhat related to time, is the serialization of parallel circuit outputs via shift register. Typical space compactors, though, refer to transforming the incoming information during a single clock cycle. For instance, because of their hardware simplicity and excellent error propagation property, parity trees can be embedded on-circuit to condense a number of probed points into a single bit stream [Fox77] [Iye89] [Rud92]. Earlier work by [Fox77] defines conditions and methods for building smaller condensation circuits using AND and OR gates. [Cha95] uses a graph colouring-based analysis to design zero-aliasing space compactors. These structures though can quickly become excessively large. Note that, in a sense, MISRs can be viewed as performing both space and time compaction.

In many contemporary circuits, the computational effort expended to attain a high quality test can be unreasonably high. Also, depending on the level of packaging, the hardware needed to apply the test can become cumbersome. The next chapter examines the concept of redesigning the circuit to alleviate these practical difficulties.

Chapter 3 Testing & Testable Designs

Testability is an abstract measure related to the cost of carrying out a high quality test. Two significant circuit attributes which reflect testability are *controllability* and *observability*. The controllability of a line indicates the ease in setting line to a 1 (or 0) by assigning values to the CUT input terminals. The observability of a line relates to the ease in propagating a line value to a primary output pin.

Testing for a circuit fault depends on the ability to control and observe a fault site. The earlier example used to demonstrate this idea (Figure 2.1) was a fairly simple and small structure. If, however, the block is embedded in a much larger cell with a small pin to gate ratio, the ability to affect the fault site may be prohibitively impaired. As a result, design for testability (DFT) is used to bring consideration of the test process directly into the design environment. The goal is to ensure that the controllability and observability of the circuit is such that a chosen test methodology (test generation, application and response analysis approaches) cost effectively achieves an acceptable fault coverage. Accomplishing this may involve the use of standardized or custom gate-level modifications, or resynthesis at a higher-level of abstraction.

3.1 Testability Estimation

As an aid to the design procedure or the decision processes of a DTPG tool, testability analysis programs have been developed to provide computationally fast estimates of the controllability and observability within a circuit. To do so, simplifying assumptions concerning expected circuit behaviour are used to devise procedures which are much less complex than those used to

generate deterministic tests. The impact of the resulting information degradation varies with the particular circuit design and the manner in which the values are interpreted and used. In all practical cases, while the measurement resolution is insufficient to indicate whether or not a specific fault is detectable, the information can usually discern a section of the CUT which is potentially difficult to test [Agr82][Hui88].

Gate-level testability estimation can be based on either structural or probabilistic analysis. Structurally derived measures usually disregard the form of input test patterns. On the other hand, probabilistic measures are dependent on signal probabilities at the circuits input lines. Probabilistic measures can be static in that they are calculated for a single input probability distribution, or dynamic in that the estimates are dependent on intermediate vector values [Kri85].

Structurally derived measures were originally used to predict test generation effort. Therefore, composed controllability and observability values reflect characteristics which might affect test generation, such as the size of the circuit [Kov79], the number of I/O pins, the number of primary inputs which must be constrained in order to force line assignments or achieve fault sensitization [Kov79], the required number of combinational node assignments [Gol79], the uniformity of an element's input/output mapping (controllability of [Gra79]), the fraction of input assignments which achieve sensitization (observability of [Gra79]) and sequential depth. Typically, at least part of the controllability and observability values for an internal line are defined by dedicated standard cell formulae. That is, given an internal line, the calculation procedure usually includes a path tracing phase in which the controllability of a node output is a function of node input controllabilities. Similarly, the observability of a node input is a function of node input controllabilites and the observability at the node output. Controllability estimation involves forward path tracing starting from the PIs, whereas observability is traced backwards from the POs. In each case the traced path terminates at the considered line.

Probabilistic representation of detection probability is important to pseudorandom testing and can be estimated from line controllabilities and observabilities. Here, line controllability refers to the probability of assigning the line to 1 (or 0) by manipulating only the PIs, and line

observability is the probability of sensitizing a path from the line to a PO. The method of [Par75] provides exact results but this is at exponential time in the worst case. In fact, the problem of computing the exact signal probability (the probability that the line is 1), thus also controllability and observability, is #P-complete [Kri86][1], therefore, approximations are devised to balance computation speed and result accuracy. Compensation for errors introduced by signal correlation due to reconvergent fanout is the source of complexity and can be a basis of distinction among methods. In the simplest, COP [Brg84], these effects are completely ignored. If the circuit/node function is represented as a cover of disjoint Boolean cubes, COP-like gate formulae yield exact signal probability results[Gho92]. Unfortunately, determining a disjoint cover is a non-trivial task. [Kri86] removes first order effects of reconvergent PI signals by using a weighted averaging algorithm based on repeated applications of COP and single bit alterations of the input distribution. A similar approach is used in the current estimation procedure of [Kri93]. There the enumerated single bit can assume any of 4-values (rather than binary) representing 1, 0 or transition states. Other algorithms attempt to reduce the effect of reconvergence by splitting associated fanouts, performing exhaustive analysis on them and using linear COP-like calculations in the non-reconvergent areas [Set85][Set86]. These procedures can provide exact results but the exhaustive analysis leads to exponential worst-case time complexity, thus, only a subset of stems is processed. Splitting of reconvergent fanout is also used in [Sav84] with the goal of propagating and computing bounds on line measures. Statistical sampling techniques based either on fault simulation [Wai85b] or fault free simulation [Jai84] have also been used to provide probabilistic measures. The accuracy of these techniques depends on the simulated test length, the manner in which fanout stems are handled and the difference in signal behaviour of the faulty and fault-free circuit.

Sequential extensions to structural analysis algorithms involve including a coefficient to reflect sequential depth. On the other hand, aside from the statistical sampling techniques, the cited probabilistic approaches only address combinational circuits at the gate-level. Probabilistic testability estimation for sequential circuits can be accommodated in the sCOP (sequential COP) method used in [Sou95a]. The procedure is based on the assumption that the circuit can be

[1] Rather than solving for the mere existence of a solution, #P-complete problems require the exact number of solutions, the verification of which requires an exhaustive search of the solution space.

modeled using Markov chain processes. However, especially considering a gate-level design, the approach can become time consuming in terms of determining the required transition matrix and solving the associated system. For gate-level approximations, a simplified version of sCOP is proposed - it is similar to combinational COP except that a time variable is associated with each line signal. Line measures are updated by iterating the calculations for a specified number of time cycles or until stabilization [Sou95a].

Evaluating sequential testability can also be done at higher-levels of abstraction. In [Che89c] [Ham91a] [Ham91b] [Kap94] [Naj92] [Gho92] [Tsu94] [Kap94] search procedures applied to OBDD (ordered binary decision dagram) representations of the circuit are used to provide a probabilistic indication of initializability, signal probabilities, and sequential controllabilites and observabilities. It is possible to account for signal correlation if the entire circuit is modeled by a single OBDD, however, this can be impractical as the size of the OBDD can be exponential in the number of circuit inputs. One solution is to partition the OBDD (i.e. segment the circuit) thus creating a number of smaller, and more tractable, subproblems. The partitioning strategy should keep the partition size small while maximizing the number of correlated nodes within each segment. Satisfying the latter criteria results in an accuracy of computation which is higher than with randomly selected partition boundaries [Kap94].

Using structural STG-analysis, the work of [Hud89] determines circuit-dependent bounds on deterministic test length and suggests such values as indicators of DTPG difficulty. Other high-level testability analysis techniques have been based on information theory definitions of signal randomness, or *entropy* [The89]. For example, if the 2^n states of an n-bit word occur with equal frequency, the entropy of the word is n . In [The89], the controllability of a module output from its input port is evaluated as the ratio of a module's input and output entropies. Likewise a global measure of controllability from the PIs is the ratio of PI entropy to that at the module output. Similarly, observability is estimated based on Mutual Information (analogous to sensitivity at the gate-level). Experiments of [McL92] finds that a serious limitation of this use of entropy is its inabililty to capture information concerning the rate at which a line switches.

Controllability and observability are both interrelated and linked to the detectability of a fault. It would be valuable to know which is more closely correlated so that efforts can be made to improve that parameter first. Based on limited experimentation on combinationl circuits, [But92] claims that observability modification could be given higher priority. For sequential circuits, this supposition does not hold. For instance, using a pseudorandom approach, experiments show that numerous lines of a sequential circuit have very low controllability - to the extent where lines may fail to switch. An appropriate assumption is that for pseudorandom-based testing, circuit controllability and observability should be at least a user-defined acceptable threshold.

3.2 Design for Testability Techniques

While there exists general guidelines for facilitating testability, such as the avoidance of redundant logic and asynchronous control paths, logic partitioning, clock isolation and memory isolation [Abr90], commitment to a formal DFT approach can vary with the type of circuit tested, design constraints and the intended method or equipment used to perform the test. For instance,

- Design specifications can limit permissible increases in circuit area, operating speed, power consumption and pinout.
- Test generation/application methodology may require DFT to assist DTPG performance (loop cutting, scan, etc.,) or reduce pseudorandom-based test length.
- Some blocks, such as memories require very different and dedicated test approaches compared to, for example, random logic.

For sequential circuits, as the controllability and observability at the state elements are typically unacceptable, many current DFT methods are based on regularized modifications which provide access to all or some of the state bits. Device-specific modifications, such as the design guidelines mentioned in the beginning of this section and the introduction of test points, may also be required.

3.2.1 Scan Design

Full scan design is probably the best known DFT technique. There are several variations of the scheme differing in latch structure, clocking, and control [Ste77] [Eic78] [And80] [Nad88] [Fun89] but the underlying principle remains relatively unchanged - by converting sequential elements (latches or flip flops) into primary inputs and primary outputs, the test problem is transformed into a less computationally intensive combinational one and direct controllability and observability of the circuit state is enabled. A good catalogue of techniques can be found in [Wil83] and [Fun89].

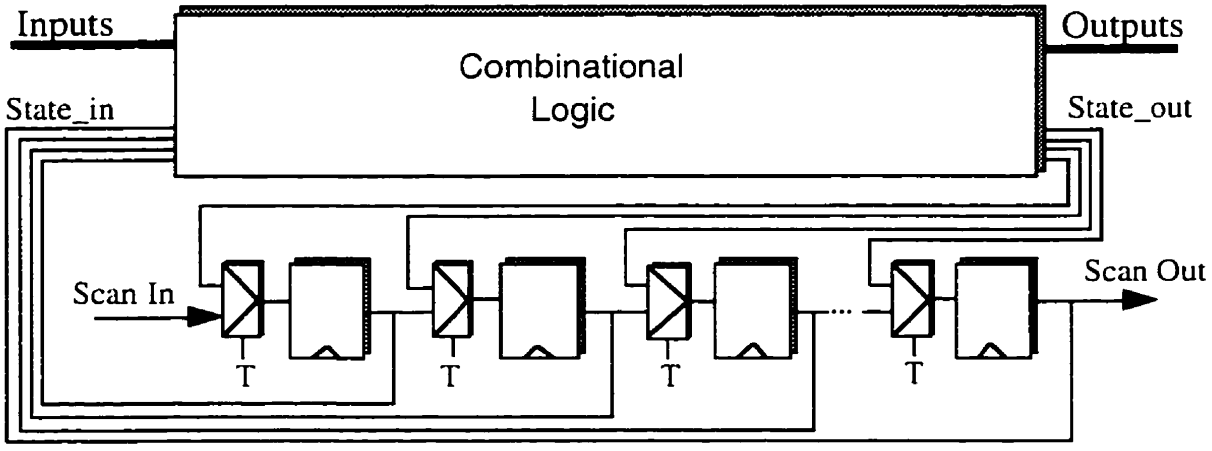


Figure 3.1: Scan Design Concept

The basic full scan design concept is demonstrated in Figure 3.1. The system as shown allows two modes of operation, *normal* and *scan*. As the name implies, in *normal* mode (T=0), the test hardware is transparent and the circuit behaves with intended functionality. In *scan* mode, the circuit latches are configured into a shift-register, or *scan chain*. During the test procedure, for each test vector, the CUT is first placed into scan mode and, aside from the bits which map directly to the primary inputs, the data is serially shifted into the scan chain. Next, the CUT returns to normal mode for one cycle after which, the scan chain is loaded with the circuit's response to the injection. Finally, the system is again placed in scan mode and the response

data is shifted out to be analysed, while another test vector is simultaneously shifted in. This method of internal access allows the flip flop outputs to be treated as pseudo-primary-inputs and flip flop inputs to be treated as pseudo-primary-outputs.

Relative to the unmodified circuit, penalties inherent to scan involve an increase in area (including that due to extra clock and chain interconnect, more complex latches and multiplexing logic), higher power consumption, and a potential degradation of operating speed. Also, serialized tests result in a test time proportional to the product of the number of test patterns and the length of the scan chain. Weighted pattern generation and store and generate approaches can somewhat alleviate the test time issue. Partitioning-based test time solutions reduce the logical length of the scan chain by selectively bypassing segments [Nar95] or introduce parallelism through the use of multiple scan chains. Realistically, the degree of partitioning is limited by tradeoffs concerning area overhead and pin limitations [Bas89]. Another somewhat contemporary difficulty arises when considering submicron technologies - wire capacitance becomes a crucial parameter affecting a flip flops drive capability. One resultant effect is that the permissible routing length between scan elements decreases as feature size is reduced. [Bar96] outlines a layout driven scan ordering algorithm which addresses the wire load issue.

Some of the drawbacks of full scan can also be addressed by including only a fraction of memory elements into the scan chain, however, in these *partial scan* systems, a sequential test problem persists. Whether or not a flip flop is included into incomplete scan chain can be heuristically ranked using structural analysis, testability measures and test generation results.

Structural techniques aim to reduce ATPG effort by picking flip flops which cut cycles, reduce sequential depth, improve initializability and minimise the number of reconvergent paths of unequal length. Testability analysis favours selection of flip flops with the lowest controllability and observability and highest sequential depth. Also, insight from test generation or switching analysis can grade flip flops according to their frequency of use in fault detection and their role in detecting faults which are relatively difficult to detect. Examples of methods which employ at least one of the above techniques can be found in [Tri80] [Agr87] [Che90b] [Pra91] [Chi90] [Gupt90] [Cha94] [Str95].

A board-level extension of chip-level full scan is *boundary scan*. There, signals at chip peripheries can be accessed via individual scan cells associated with each of the primary input and output pins. The gains of this approach are especially significant when considering high density single or dual sided surface mount boards [Par89][Gre94]. Compared to conventional bed-of nails testing [Gra89], electrical node isolation is more reliable than physical probing. In addition, depending on circuit/board density, size and positioning of probe pads can become expensive and cumbersome [Gre94]. Boundary scan eliminates device overdriving, and hierarchical design for testability is facilitated.

3.2.2 Non-Scan DFT

In certain instances, such as those requiring tight area and performance expectations or low pseudorandom test time, scan can be rendered unfeasible. In attempting to address this, some techniques accept the higher complexity of sequential test generation and test the CUT without the use of scan. In most cases the aim is to test the circuit using test patterns applied in parallel to all PI pins - similar to the natural input access of the circuit. Compared to scan, if test application is done at/near the normal operating speed of the circuit, this input data format is more effective in detecting unmodelled faults. Also, more patterns can be applied in the same test interval.

Rather than building a serialized scan chain, [Chi93] (and similarly [Lee90]) permits at-speed parallel application of DTPG tests by selecting a portion of flip flops for direct individual access from existing primary input pins. The criteria for flip flop selection are similar to those used for partial scan. Although high fault coverages are reported, due to introduced dependency among flip flops, the effectiveness of the method may degrade as the number of loaded flip flops surpasses the number of primary input pins. Of course, the introduction of additional PIs can offset the dependency problem but this solution can become unacceptably expensive. A similar idea to that of [Chi93] is used in the pseudorandom testing method of [Sou95b]. There, each flip flops is modified by feeding its normal input through a 2-input XOR. In normal mode, the additional XOR is transparent, and during test mode, the second XOR input is logically

connected to a PI which provides an equiprobable pseudorandom signal. The number of additional PIs is kept manageable by grouping the flip flops into subsets such that the range of activity due to each member is mutually independent. In both methods cited, the global pin to flip flop routing can introduce an intolerable area cost. Also, as the full state controllability of scan is retained, the classification as *non-serial-scan* is probably better suited to these schemes. Random Access Scan Design [And80] is another non-serial-scan architecture. There each state latch is uniquely addressable, much like a random access memory. The area and performance penalties may be high with this approach due to the sizable addressing logic.

Knowing that set/resettable flip flops are smaller than scan cells, in the *partial reset* approach, flip flops crucial to initialization are modified to include a direct set or reset capability which is usually controlled by multiple global control lines [Abr93] [Mat93] [Sou95a]. Again, partial scan techniques can be used to select elements to be altered. While the method was originally proposed for DTPG applications, given that resetting a subset of circuit flip flops can permit a wide range of non-unique reset states, recently this method has been used to enable pseudorandom testing of sequential circuits[Sou95a]. There flip flops are selected in two passes: first choosing those elements which best aid initialization, and second by targeting those which address the detectability of fault groups.

The at-speed non-scan implementation [Mur95a-b] recognises that a large problem encountered in pseudorandom sequential testing is poor accessibility of the flip flops which manifests itself in a very low switching rate at these elements. The conjecture is that since each applied random pattern causes activity which is potentially a sub-task needed to detect a sequential fault, and application of a subsequent generated test pattern tends to nullify this partial work, the challenge is to modify the CUT and generation scheme so that the activity introduced by each applied pseudorandom test pattern can be stored for future use and so that the number of circuit states attainable using pseudorandom patterns is increased. Accomplishing these tasks suggests more efficient usage of a pseudorandom test pattern.

The approach uses the natural sequential functionality of the CUT to introduce a larger spread of circuit states than that usually possible if pseudorandom-based patterns are applied to an

unmodified CUT. This is done by altering a partial set of flip flops so that either a rarely occurring 1 or 0 can be selectively retained. The resulting *trap* cell output bias is gradually forced towards a near equiprobable value, thus the internally generated sequence of biased test patterns contains "semi-random" patterns (or "semi-random states") embedded at various intervals.

The internal generation process probes and retains the effect of less frequently occurring circuit activity which can be related to the effect of rarely activated fault sites. Such use of existing circuit memory can also amplify an occurring fault effect by enabling repeated attempts to propagate captured activity to an observation point, an unmodified flip flop or another Trap Cell. Moreover, just as some algorithmically generated test blocks are alike in that they set up identical or low Hamming distance circuit states, the trapped bits of an internal pattern hold partial circuit states while allowing the PI's and the rest of the internal pattern (and a number of subsequent internal patterns) to "search" these states for faults. The slow rate of change of internal state is similar in effect to the test generation algorithm of [Abr92] in which each new circuit state is frozen and followed by a combinational test on the resulting logical sub-network. In fact, since internal patterns attributed to trapping reflect circuit activity, and thus possibly at least a partial propagation of a fault effect, the constructed semi-random internal test patterns are potentially more useful than externally injected "shot in the dark" equiprobable test patterns. Although not considered in [Mur95a-b], observability modifications can further enhance fault detection. The DFT implementation presented in Chapter 4 can also be used to eliminate the globally routed control signals which regulate the trap function.

In all of the above non-scan approaches, the sequential test problem is eased by altering the function at the memory elements. However, as in scan, the rigidity of modification sites can impact the applicability of the techniques, for example in performance-constrained designs. Ad-hoc DFT such as test point insertion can remove this restriction while offering a comparatively smaller area overhead, or augment the above systems themselves.

3.2.3 Test Point Insertion

The locations of circuit modifications in test point placement are device specific. The principle of test point insertion has existed for some time [Hay74] and the optimal placement problem is known to be NP-complete [Kri87]. Recently though, the subject has been revisited. Classified by the intended impact of the test points, there are two categories of approach. In the first, similar to methods used to select partial scan locations, test points are inserted to aid DTPG [Gun90]. The second family of solutions facilitate hardware-pattern-generation-based testing by either segmenting the CUT to enable pseudoexhaustive testing [Wun89b] [Wu92] or by enhancing pseudorandom testability [Bri86] [Iye89] [Sav89] [Sei91] [Sav91] [Lin93] [Tou96] [Mur96]. In the pseudoexhaustive case, however, the area overhead and speed degradation can become large if the test application time is reduced to within a practical range [Lin93]. Furthermore, in both pseudoexhaustive and pseudorandom scenarios, inclusion of full or partial scan is usually required.

Considering pseudorandom pattern testability, a number of recent test point insertion schemes [Bri86] [Iye89] [Sav89] [Sei91] [Sav91] [You93] [Tou96] have been developed in the context of combinational circuits while only a few, such as PBIST[Lin93], deal with sequential ones. These approaches usually use fault simulation or testability analysis to guide test point placement. For instance, in [Bri86], fault simulation results and analysis of reconvergent nets are used to insert test gates which reduce signal correlation at gate inputs. The system described in [Iye89] extends this work in the context of using scannable test latches but, rather than monitoring correlation, targets gates which block the propagation of faults not detected by simulation.

Test point methods based on probabilistic estimation of testability measures sacrifice the accuracy of fault simulation for speed. For example, in [Sei91], testability information derived from COP [Brg84] is used in conjunction with a cost function and a gradient calculation method adapted from [Lis87] to reflect the global impact of a test point candidate on the pseudorandom pattern testability of a scan circuit. With reference to circuits tested with a type of circular self-

test path[Kra87], [Sav89][Sav91] also use COP to identify and rank a number of candidate test point sites. Additional heuristics outlined in [Sav91] [You93] address the placement problem using the concept of *fault sectors* - a fault sector resembles a fan-in or fanout subnetwork containing a group of hard to detect faults which can be affected by a single insertion at the sector origin (a fan-in node for an observability point and a fanout node for an controllability point). In [Lin93], an algorithm similar to [Sav91] determines test points which augment sequential BIST of near acyclic partial scan circuits. Instead of using COP, the authors derive probabilistic measures applicable to this class of circuit. As an alternative to test points, the authors also suggest the use of deterministic testing to cover the random pattern-resistant faults remaining after pseudorandom tests are applied. However, in a BIST platform, the latter option can become self-defeating since the size of the stored 'reduced' test set can approach 70% of the complete deterministic test [Bas89]. The work of [Che95] revises the efforts of [Sei91] and [Lin93] to also consider performance impact as an insertion criteria.

In [Tou96], faults are injected into a combinational circuit and path tracing is used to determine modification sites. Rather than introducing additional scan cells, gated signals from existing scan cells or primary inputs are used to drive the test points. Although there is no comment on the area overhead consumed by this structure, it is suspected to pose similar difficulties as the non-scan DFT methods discussed in Section 3.2.2. The method introduced in Chapter 4 outlines a manner in which a different approach to path tracing can be used to insert a non-scan-based test point network.

3.2.4 Built-In Self-Test

One of the currently important classes of DFT involves the integration of ideally all test circuitry onto the CUT itself. A standard setup of such built-in self-test (BIST) systems is shown in Figure 3.2. The additional test execution blocks shown are transparent to the normal-mode function of the device. Testability enhancing hardware, such as scan or test points may also exist.

During test mode, input stimuli are accepted from the test pattern generation block (TPG) and the CUT's output responses are prepared for analysis in the output compaction block (OCB). At the end of the test session, a final signature is compared with reference values and a result flag is signaled. [Lam95] has shown that fault simulation of BIST systems can be conducted more efficiently if multiple intermediate signatures are sampled.

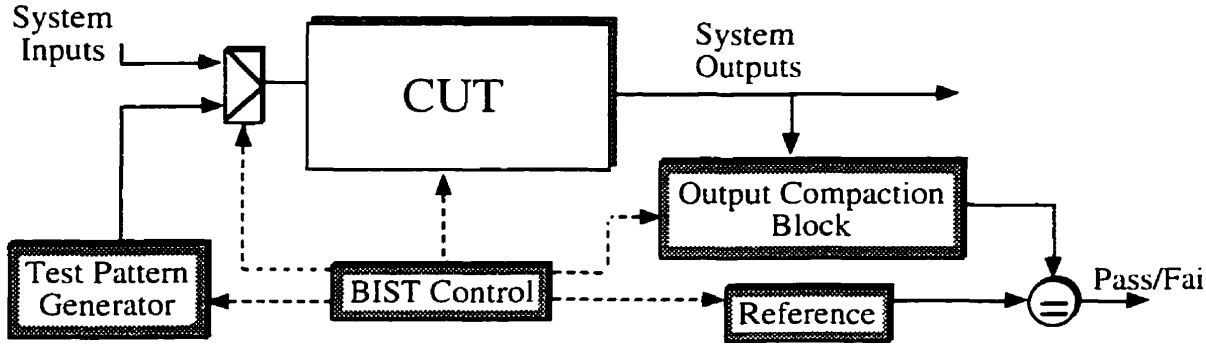


Figure 3.2: Standard BIST Scheme

A *distributed* BIST architecture is one in which each logic core is tested with dedicated TPG and OCB blocks, while a *centralized* BIST structure utilizes a single TPG and OCB for multiple cores. Because of the increased parallelism, and as the TPG and OCB can be customized, distributed BIST can attain higher fault coverage and lower test times than centralized BIST but at higher size, power and possibly performance costs. Other factors which impact the choice of architecture per system segment are the level of packaging, the intended size of replaceable units (ideally each unit should be self-testable) and the compatibility of test approaches versus targeted logic.

Pursuing self-test option is ultimately to reduce test costs, and compared to conventional off-chip approaches, there are indeed many potential advantages. For instance:

- The required complexity of external test equipment can be dramatically reduced.

- Diminished dependency on probes access leverages the test procedure to various levels of packaging [Bar87][Par89][Gre94] and can allow lower long term maintenance costs (e.g. user-initiated in-field tests for microprocessors [Kub83]).
- Potentially shorter test time in cases where partitioning is employed and/or tests are run near circuit speeds[Kra87]
- The random test patterns often used in near operating speed tests tend to detect many unmodeled faults [Max91] [Dea94]

However, there are hurdles encountered attempting to achieve these benefits, including :

- The additional area overhead of BIST circuitry can lower device yield.
- A potential for performance degradation if test circuits are included in a critical path.
- The generated test lengths may require dedicated DFT to minimise test time costs
- There is a general lack of knowledge and design automation to address the above issues.

Fortunately, none of these problems are insurmountable. Proponents of BIST claim that the long term benefits offered by this test option far outweigh the drawbacks - in some cases even if the overhead approaches 40% [Tot88]. Furthermore, self-test may be the only solution to test access limitations which arise from increasingly dense chip and board-level packaging [Par89][Gre94]. Numerous introductions to BIST have been published, some can be found in [McC85a][[McC85b][Bar87][Agr93].

3.2.4 1 Test Application Format in BIST

The TPG block is usually implemented using a hardware pattern generation approach. In cases, such as store and generate schemes or programmed tests, the memory required should be compact enough to conform to the tolerable implementation overhead or reside off-chip. Since the sequel is confined to pseudorandom testing of sequential random logic, the outlined on-chip TPGs are based on variations of pseudorandom generators. Categorized by the input data format and the rate at which the CUT accepts and responds to this stimulation, BIST procedures can be executed in a *test per scan load* or *test per clock* manner.

3.2.4.1.1 Test Per Scan Load

As explained earlier, scan design concepts serve as the basis for many DFT schemes, including those used for BIST. A test per scan load process is similar to the generic scan operation outlined in Section 3.2.1. Here, the bulk of the test time is consumed in loading and flushing the scan chain, thus methods such as weighted generation and store and generate techniques may be used to reduce test length.

Test per scan load which employ full scan pertain to combinational testing [Bar82] [Bar84] [Eic83] while partial scan based systems refer to sequential mode tests [Lin93].

3.2.4.1.2 Test Per Clock

In test per clock methodologies, the CUT accepts and responds to test data after each clock cycle. Compared to Test per scan, it is more difficult to design these systems, however, as testing can be performed at-speed, test times can be orders of magnitude smaller. Test per clock schemes may require the use of a number of scan chains or be free of scan.

In scan-based designs, flip flops are usually grouped into enhanced single-bit or multi-bit registers which, in test mode, serve as test pattern generators or response compactors. In certain configurations, a register can serve both purposes while in other situations these functions must be exclusive. The serialized access of the scan configuration is used primarily for initialization and removal of the internal test signatures.

One of the milestone structures proposed for scan-based test per clock BIST is a multifunction shift register called a *Built-In Logic Block Observer* (BILBO) [Koe79]. The BILBO, and its cellular automata counterpart, CALBO (cellular automaton logic block oberver [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 analyser. Figure 3.3a is a typical BIST setup using a BILBO. Circuit blocks A and B share the BILBO for different purposes and are tested in separate test sessions - in the first, the BILBO acts as a compactor for circuit A, and in the second, it acts as a pseudorandom TPG for B.

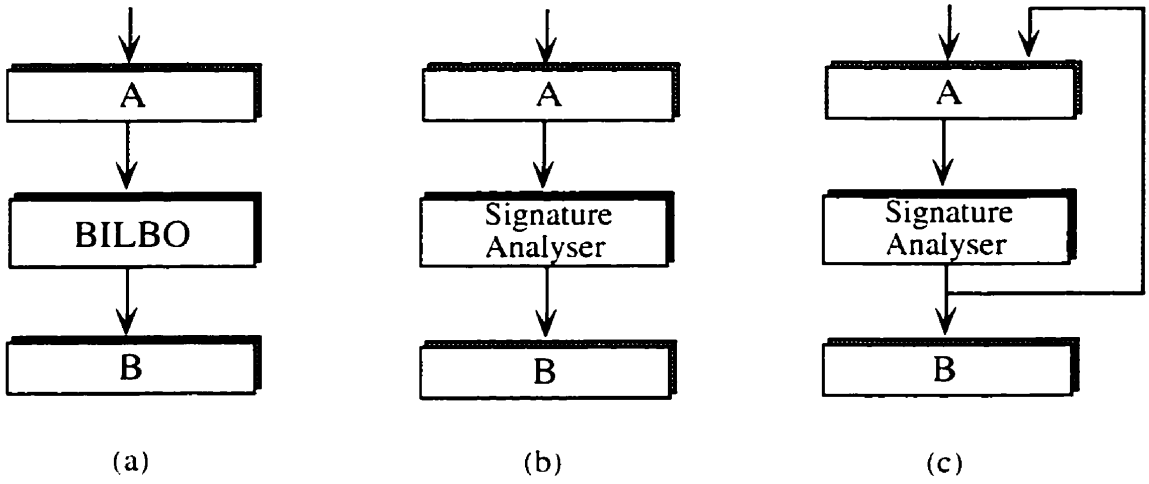


Figure 3.3: Partitioned BIST - (a) BILBO register tests A &B separately; (b) signature analyser register configuration allows parallel testing of A&B; (c) A&B cannot be tested in parallel because of feedback dependence.

Compared to the above procedure, sometimes a test time reduction and simplification of BIST control logic and register structure becomes possible through the use of signature registers as pseudorandom TPGs [Kim88]. The idea is demonstrated in the modified setup of Figure 3.3b in which the each signature from A is used as a test pattern for B. Thus now, A and B are tested in parallel. In such implementations, the pseudorandom properties of the signature analysers are not guaranteed if there exists feedback from the analyser to the logic for which it is a compactor (Figure 3.3c). In these situations, it may become necessary to partition the cycle using additional BILBO-type registers which are transparent in normal mode. Alternatively, a BILBO structure within a cycle can be modified to a L3-BILBO [Gup81] or CBILBO [Wan86] (i.e. internally partitioned using additional latches) so that it can generate patterns and compact responses independently. As the implementation of these "fixes" can impose considerable hardware costs, [Str95] outlines a cycle-breaking optimization technique which selects and inserts BILBOs and CBILBOs so that each circuit loop contains at least one CBILBO cell or two BILBO cells. Compensating for the effect of feedback can also be handled as part of a state encoding problem which results in an increased spread of output states at the compactor [Chu89] or a reduction in the number of cycles within the network [Che89d]. State encoding is

also examined in [Esc90] with the intent of implementing the entire sequential circuit as a modified MISR.

The simultaneous use of flip flops as TPGs and OCBs is also suggested in full scan methods, such as *simultaneous self-test* (SST) [Bar87], and partial scan approaches based on the popular *circular self-test path* (CSTP) [Kra87], both of which were originally proposed for non-partitioned logic. In many experimental cases the use of these systems result in acceptably high fault coverage. However, in SST the degree of randomness and extent of error cancellation can be difficult to predict [Abr90]. Furthermore, although CSTP can provide good pattern coverage leading to high fault coverage in a limited number of clock cycles [Abr90][Sla92], existence of circuit loops can inflict severe detrimental effects on test results since the state space traversed can be excessively narrowed. Based on STG analysis, one approach to avoiding premature repetitions of generated patterns involves selection of an appropriate initial state which is a maximum distance from network loops [Bry90][Cor94]. Unfortunately, the extent of the solution can be limited by the connectivity, availability and size of the STG. Preprocessing using the aforementioned state encoding methods of [Che89d] and [Chu90] may also be helpful to this end.

Non-scan test-per-clock approaches range from rudimentary (and relatively ineffective) *cyclic analysis testing*, where PO signals are multiplexed with PI signals [Abr90], and *centralized and separate board-level BIST* [Ben75], to more sophisticated concurrent architectures [Sal88] and the non-scan techniques already discussed in Section 3.2.2. An original non-scan test per clock BIST scheme is introduced in the next chapter. There, unlike all of the above approaches, the required modifications are not localized to state elements. This however, is just one manner in which the new system departs from these conventional architectures.

Chapter 4 Self-Driven Sequential BIST

This chapter examines a new gate-level non-scan test per clock DFT technique called *Self-Driven BIST* (SD-BIST) that permits at-speed pseudorandom testing of resettable synchronous sequential circuits. The foundation of the scheme is an original test point implementation which is unique in that, apart from a test mode flag, all I/O signals required for test point operation are tapped from nodes already existing within the circuit. It is in this sense that the system is deemed *self-driven*. This property virtually eliminates the control signal generation hardware typically needed to operate conventional test point-based structures.

When placed in test mode, at the normal system clock rate, the circuit is tested using parallel pseudorandom patterns applied only to the circuit under test's (CUT's) primary input pins and analyzing or compacting the corresponding responses at the primary outputs (POs)[1]. Along with the advantages related to test time reduction and potential unmodelled fault detection associated with this test format, other benefits of SD-BIST over scan-based approaches include the elimination of additional test clocks, reduced test cell size and the potential for lower performance degradation. Furthermore, many of the concepts involved in designing self-driven test hardware are applicable to other BIST strategies supported by test points. Conversely, while test point placement methods are introduced in later sections, the self-driven implementation remains valid regardless of the particular means used to determine the modification sites.

[1] Without loss of generality, the input signal probabilities follow an equiprobable distribution.

Subsequent sections discuss the general SD-BIST concept, simple fault free testability estimates which guide the modification process, a path tracing procedure used to identify test point locations and test network implementation. At various stages experimental results are offered in order to help demonstrate the effectiveness of the new procedures. Development considers the single stuck-at fault model, and as a simplifying assumption only, the maximum fanin of a logic gate is 2.

4.1 Test Points for BIST

The previously discussed test point schemes used for pseudorandom testing (section 3.2.3) require inclusion of full scan or partial scan. Moreover extra hardware (comprised of, for example, LFSR bits added to the pattern generator, scan cells, test clock and control pins, etc.,) separate from normal functional logic is needed to generate control signals necessary to the operation of controllability points. Such complex control logic is not used in SD-BIST.

4.1.1 The Self-Driven BIST Concept

In test mode, analysis of fault free information local to a circuit line can be used to quickly recognize regions that might hinder fault detection. For example, the existence of a number of fixed polarity lines indicates areas of poor controllability which narrow circuit activity and limit fault coverage. Similarly, gate inputs with a sensitivity, or *one-level sensitization probability* [Jai85] (i.e. the probability, $\mathbf{OL_i}$, that the bit value, **i**, at that gate input is observable at the gate output), equal to zero suggests poor error propagation along paths containing those lines.

In this method, local fault free information is compiled into a *switching profile* which records line *biases* (i.e. the probability that the line is logic 1), and the 1 and 0 observability values local to a gate ($OL_i \mid i \in \{0,1\}$). It is known that prudent insertion of controllability points can adjust CUT activity so that, in test mode, ideally each circuit line switches and the OL_i of each gate input is at least a threshold value greater than zero. This establishes a minimum requirement for high pseudorandom pattern fault coverage. The improvement suggested by SD-BIST is to

replace the controllability input signals, which are usually derived external to the circuit's function, with local feeds from existing internal lines. Compared to existing test point schemes used for pseudorandom BIST, this modification simplifies the required control signal generator to the complexity of wires, thus the relative size of the CUT is decreased (figure 4.1). In addition, compared to the restriction to simple equiprobable or coarsely weighted control probabilities common to conventional schemes, internally probed control signals provide a "richer" set of almost arbitrary control signal probabilities. Thus, there now exists an extra degree of freedom when manipulating the controllability of the circuit.

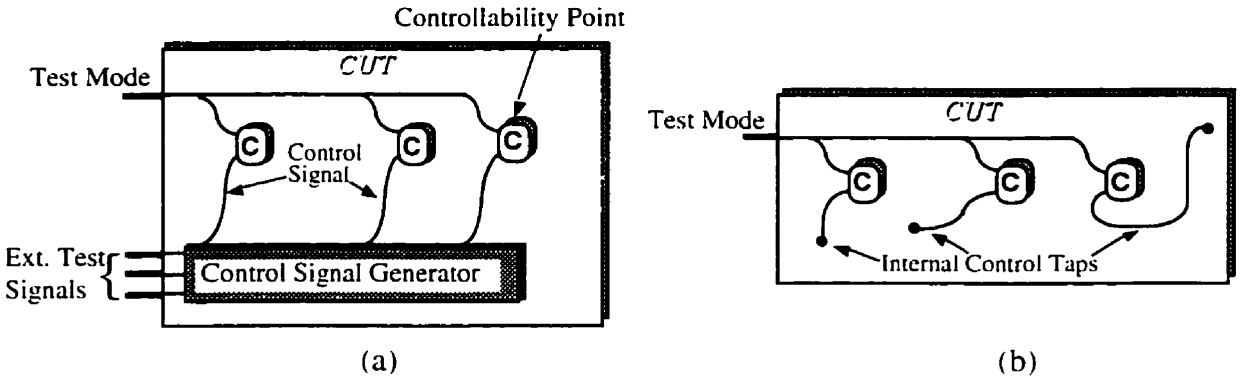


Figure 4.1: Comparison of Control Hardware Size
(a) Conventional BIST - Control is derived external to circuit function;
(b) SD-BIST Controllability Point - Control tapped from circuit function.

After an acceptable fault free switching profile is established, a further rise in fault coverage is gained by increasing the observability of groups of yet undetectable faults. As shown in Figure 4.2, the observability points of SD-BIST re-route error signals to internal sites from which, in one cycle, the errors are retained at the state elements or the respective faults become detectable at the primary outputs. Furthermore, by selecting an internal site which is unaffected by the considered fault set (the source of the errors), fault free analysis can be used to estimate the observability of the transferred error at the POs or state elements. Indeed, observability point outputs can be restricted to flip flop inputs or sites directly sampled by the test analyzer, however, the opportunity for such connections may be limited by factors, such as the amount of reconvergence tolerable at a line before adversely affecting fault coverage, the timing penalty

caused by gating a number of signals to a single line, pin count and the size of the signature analyzer. Therefore, the broadened solution space offered by the proposed structure is beneficial.

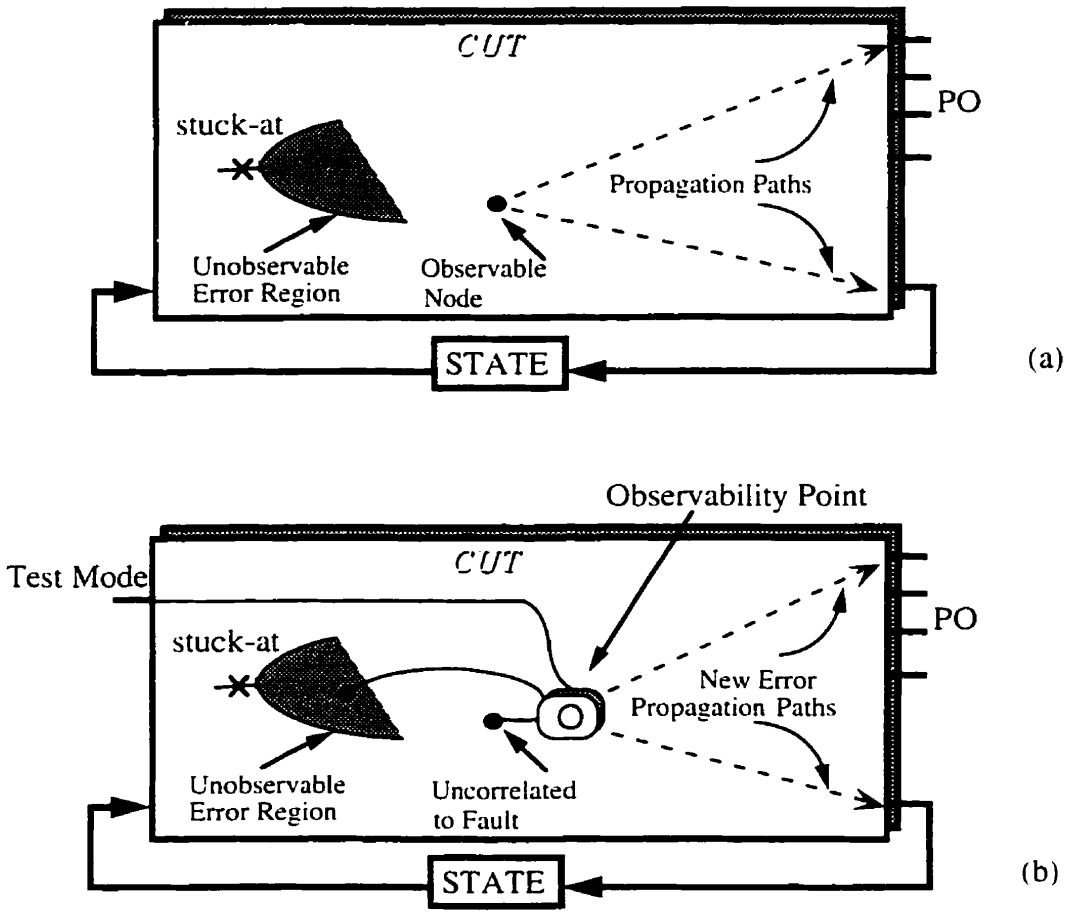


Figure 4.2: Observability Point Operation - (a) unobservable fault exists before observability connection; (b) error stream is branched to reach either PO or state variable

Based on the preceding concepts, the design of self-driven test hardware consists of three steps:

1) Insertion of externally driven controllability points to achieve a near ideal switching profile.
2) Replacement of externally driven controllability point inputs with internal signal taps
3) Insertion of observability points.

In steps (2) and (3), correct circuit modifications are those which maintain the predictable switching established in step (1). Required structural circuit data is retrieved using straightforward path tracing, and sufficient functional information for controllability alteration is approximated using fault free signal probability estimates and fault free logic simulation. Fault domain analysis is introduced only when determining observation points. User-defined thresholds related to variables, such as signal probability, observability and error probability guide the trace routines and aid in selecting lines which can be used to build the self-driven structures.

4.2 Testability Estimates

Focusing on circuit switching allows much of the circuit analysis to be done in the fault free domain thus reducing computational effort. Measurements used throughout the scheme are: line biases, local and global observabilities, and the propagation of signal probability disturbances within the system. Initial reference values for line biases and OL_i are sampled from logic simulation.

4.2.1 Fault Free Observability

The pairwise global observability of an event at a line **i** seen at **j** is:

$$OBS_{ij} = \sum_{k=0}^{n} obs_{ij} \cdot t^{k} \qquad \text{Equation (4.1)}$$

Due to the existence of storage elements, the observability is not a single lumped value but interpreted as a number of observabilities, $obs_{ij}\, t^k$, each associated to a specific time frame, **k**. The estimation procedure uses sampled OL_i in a circuit tracing method similar to STAFAN [Jai85]. $obs_{ij}\, t^0$ is calculated as a weighted sum of the product of local observabilities along each path backtraced from **j** to **i** and concludes when a flip flop is reached. Observabilities for subsequent time frames are computed along paths traced starting from each flip flop

encountered in the previous time frame. Therefore, the upper bound on the number of time frames is equal to the sequential depth of the longest path from **i** to **j**. Since sequential loops can exist, the number of time frames per calculation is limited. In the event that multiple paths are traversed during a backtrace (due to reconvergence), either the maximum path value or a real-valued OR of all traced observabilities can be retained at the associated stem. An error coefficient can also be included at this point [Abra90][Jai85].

4.2.2 Bias Offset Propagation

The impact of an externally driven test point is considered equivalent to injecting a disturbance in signal probability and rippling its effect through the circuit. Such a bias offset, Δ, possesses sign and magnitude - a positive sign indicates a rise in signal probability while a negative sign denotes a reduction. Calculations are based on standard signal probability formulae for 2-input gates [Bar87] [Brg84]. As the procedure is rather straightforward, for brevity, it is demonstrated for a 2-input AND gate only and formulae for additional logic gates are provided in appendix A. The new output probability, $P_{disturbed}$, given propagating offsets Δ_1 on input-1 and Δ_2 on input-2 is shown below:

$$P_{original} = P_1 P_2$$
$$P_{disturbed} = (P_1 + \Delta_1)(P_2 + \Delta_2) = P_1 P_2 + P_1 \Delta_2 + P_2 \Delta_1 + \Delta_1 \Delta_2 \omega$$
$$\Delta_{out} = P_{original} - P_{disturbed} = P_1 \Delta_2 + P_2 \Delta_1 + \Delta_1 \Delta_2 \omega \qquad \text{Equation (4.2)}$$

Here, Δ_{out} is the bias offset propagated to the AND gate output, and P_1 and P_2 are the respective input pin biases. All calculations assume that offset injection occurs at a single site and the Δ at a gate input is independent of the bias at the neighbour input. A non-zero $\Delta_1\Delta_2$ term suggests reconvergence of the injected offset. Since the components of this term are not independent, a correction factor, ω, is introduced. ω can be the ratio of the sampled probability of the input bit pair [i,j] affected by the $\Delta_1\Delta_2$ term versus the probability expected using P_1 and P_2. The bit pair in question depends on the gate type and the sign of Δ_1 and Δ_2. For example, considering a 2-input AND gate with a positive Δ at each input, the relevant case

is [0,0]. In general, let S[i,j] be the sampled probability of the affected bit pair [i,j] at inputs 1 and 2 of the 2-input AND gate, then, ω can be defined as in table 4.1:

Δ_1	Δ_2	ω
>0	>0	$\frac{S[0,0]}{(1-P_1)(1-P_2)}$
<0	<0	$\frac{S[1,1]}{P_1 P_2}$
<0	>0	$\frac{S[1,0]}{P_1(1-P_2)}$
>0	<0	$\frac{S[0,1]}{P_2(1-P_1)}$

Table 4.1: Correction Factor ω for 2-input AND

Unfortunately, these error corrections may lose meaning as the offsets cycle through the state elements. That is, after each time frame, shifts in phase of Δ_1 and Δ_2 can cause them to appear less correlated. Conversely estimation errors can become compounded if they are allowed to traverse a number of time cycles. However, in most experimental cases, the assumption of signal independence (ω=1) was satisfactory.

Using Equation 4.2, an example of offset propagation is presented in Figure 4.3. Figure 4.3a is the original circuit segment with line biases shown. Assuming that a test point injects an offset $\Delta_2 = 0.72$ at input-2 of AND-A, Figure 4.3b shows the propagation of the disturbance and the updated biases. Δ subscripts are omitted for convenience.

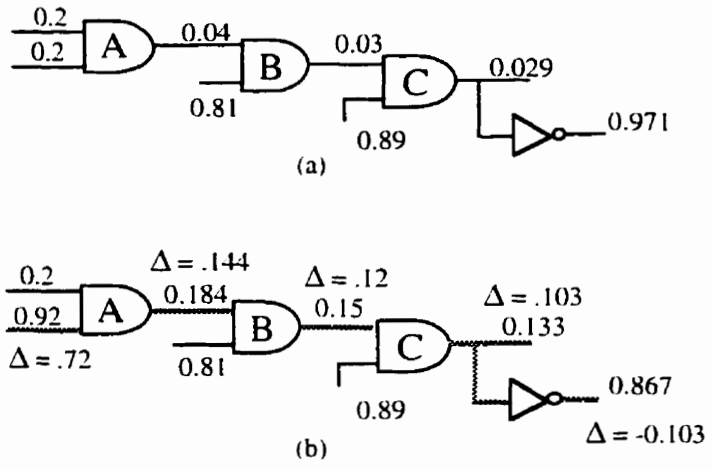


Figure 4.3: Example of Offset Propagation Needed to Model Test Point Insertion

In the event that the test point causes an inversion of a gate input, Δ_{out} is calculated as:

$$\Delta_{out} = \gamma_1\gamma_2 P[0,0] + \gamma_1(1-\gamma_2)P[0,1] + \gamma_2(1-\gamma_1)P[1,0] - (\gamma_1+\gamma_2-\gamma_1\gamma_2)P[1,1] \qquad \text{Equation (4.3)}$$

where γ_k is the probability that the input k is inverted and P[i,j] is the probability (found via sampling or otherwise) of the bit pair [i,j] at inputs 1 and 2 of the AND. For instance in Figure 4.4, if an XOR is used to invert the second input of gate A with a 5% probability ($\gamma_2 = 0.05$), only bit pairs [0,1] and [1,1] can change to affect the output bias Also, assume that the normal inputs of A are correlated so P[0,1] = 0 and P[1,1]=0.5. Thus, by Equation 4.3, Δ_{out} = -0.025.

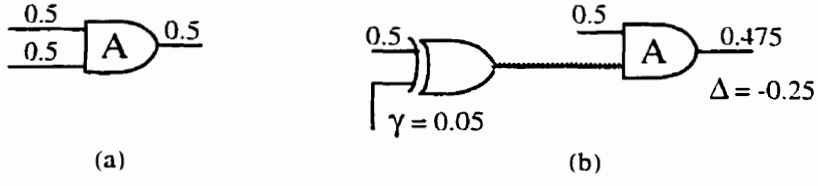


Figure 4.4: Offset Propagation for Inversion

Over a number of time frames, the injected Δ value is propagated to all affected gates and flip flops. As the Δ at all elements may not reach a steady state value, the depth of the calculation is

limited. In the event that a predicted signal probability exceeds 1.0 or drops below 0.0, $P_{disturbed}$ is forced to the respective upper or lower limit.

4.2.3 One Level Sensitivity Estimation

After bias offsets have been determined, assuming that all signals input to a gate are independent, updated OL_i at a gate input can be approximated as the non-controlling bias at the neighbour pin. However, since sampled original signal probabilities and OL_i values are available, alternate estimates can be found. For instance, the normalized OL_0 at pin 1 of an AND gate is (the relevant bit pair is [0,1]):

$$OL_0 = (1-P_1)P_2 / (1-P_1)$$

$$OL_{0\text{-new}} = (1-P_1-\Delta_1)(P_2+\Delta_2) / (1-P_1-\Delta_1)$$

$$OL_{0\text{-new}} = (P_2- P_1P_2+ \Delta_2-P_1\Delta_2- P_2\Delta_1- \Delta_2\Delta_1)/(1-P_1-\Delta_1) \qquad \text{Equation (4.4)}$$

Here, the sampled value for P_1P_2 is known. By enumerating all possible input bit pairs, Equation 4.4 can also be expressed exclusively in terms of sampled OL_i and Δ's.

The sampled or estimated probabilities used in the calculations of this section characterize a history of, possibly unacceptable, CUT behaviour *before* an offset is injected. Since it can be difficult to predict the relationship among newly switching lines caused by a Δ injection, estimation errors can be introduced which, as mentioned, can become worse as the Δ cycles though feedback loops. For instance, correction values calculated in table 4.1 can become invalid if $\Delta_1\Delta_2$ exist but the associated lines did not switch in the pre-injection reference switching profile. Also, due to reconvergence, the Δ at a gate input may be correlated to the bias at the neighbour input, thus violating the simplifying assumption of signal independence. Fortunately, experimental results indicate that the approximations made in the above formulae do not result in the failure of the modification procedure. This is because the measures are intended as a crude guide to evaluate the relative performance of various changes to the circuit.

4.3 Controllability Enhancement

The adjustment of circuit switching is a two phase process. Since self-driven controllability points may interact, while determining the complete set of modification sites, all controllability inputs are driven with independent externally derived signals. After all points have been placed, the external test sources are systematically replaced with internal ones. The desired result is a near ideal switching profile. In this section, all analysis for controllability enhancement is fault-independent.

4.3.1 Externally Driven Test Points

Iterative insertion of externally driven controllability points progressively alters line biases such that each insertion incrementally improves the switching profile.

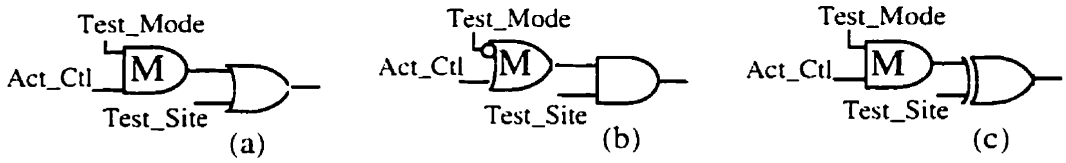


Figure 4.5: Controllability Points to (a) inject an increase in signal probability - OR-type; (b) inject a decrease in signal probability - AND-type, (c) invert a component of Test_Site signal probability - XOR-type

The three types of controllability point cells used for experimentation are shown in Figure 4.5. Here, Test_Site corresponds to the original circuit line being modified, thus the cell output is connected to the circuit node originally fed by Test_Site. At the mode-select gate (M), Test_Mode toggles between normal and test operation while the controllability input, Act_Ctl, regulates the bias injection at the cell output. Let TS be the Test_Site bias and INJ be probability of injection calculated by gating the respective Act_Ctl and Test_Mode signals. Then, governed by the formulae of table 4.2, the cell of Figure 4.5a injects an increase in signal probability, Figure 4.5b injects a decrease in bias and Figure 4.5c inverts a component TS.

Controllability Point	Injection Formula
OR-type	Δ = INJ (1 - TS)
AND-type	Δ = - TS (1 - INJ)
XOR-type	Υ = INJ

Table 4.2: Offset Injection Formulae for Controllability Points

Figure 4.6 is an iterative procedure for insertion of the controllability points where the Act_Ctl input is driven from a source external to the CUT.

```
1) Generate initial switching profile via logic simulation
2) Backtrace and collect test site candidates
3) Rank candidates using offset propagation
4) If all candidates fail rank
      4a) Change trace thresholds --> go to (2)
5) Insert the externally driven test point possessing best rank
      5a) Estimate post-insertion switching profile
      5b) if reference switching profile not improved
            5c) Try next best candidate --> go to (5)
         else
            5d) reference profile = post-insertion profile
6) If switching profile is near ideal
      6a) DONE  (--> proceed to self-driven implementation)
   else
      6b) Go to (2)
```

Figure 4.6: Summary of Externally Driven Activity Point Insertion

Given an initial reference switching profile determined using logic simulation, a set of Test_Site candidates is accumulated using a threshold-driven backtracing technique in which flip flops are transparent. This begins by identifying all lines with an OL_i value below a user defined *block threshold* (i.e. *blocking* lines). These lines are assumed to hinder error propagation and circuit switching. Next, since the OL_i at a gate input is influenced by the bias at the other inputs to that gate, for each blocking line, the circuit is backtraced along the most controlling path commencing at the neighbour of the blocking line and terminating when an OL_i value above a

user-defined *stop threshold* is reached. As a result of this search procedure, a controllability point positioned at an extracted Test_Site has the global effect of adjusting the switching of a number of blocking gates traversed along the associated path.

For each backtrace, the appropriate controllability point can be determined knowing the start gate and the number of inversions encountered. Table 4.3 lists the cases for AND-type and OR-type cells.

Start Gate	Inv. Count	Controllability point
AND/NAND	even	OR-type
	odd	AND-type
OR/NOR	even	AND-type
	odd	OR-type

Table 4.3: Controllability Point Selection

For AND-type and OR-type points, the line at which the trace halts is the Test_Site candidate. When XOR-type cells are selected, the output of the last gate traversed is returned. Inversion (XOR) is always valid but it may be required to limit this option since XOR-type cells are larger and slower. Conditional selection of XOR-type cells is based on whether the OL_i at the stop line exceeds a threshold *xor-select* value . The choice of a high threshold (e.g. 0.9) implies a significant change in bias at the Test_Site without disturbing the high OL_i at the stop line and its neighbour. Suitable Act_Ctl and threshold values are proposed in Section 4.3.3.

An example of Test_Site extraction is shown in Figure 4.7. Each line is labeled with its bias. Assume that the OL_i block threshold is set to 0.005, the stop threshold is 0.1, xor-select is 0.9, the circuit is always in test mode and each pair of gate inputs is uncorrelated. Then, the OL_i at a gate input is equal to the non-controlling bias at the neighbor pin and backtracing initiates at gates C and F. The trace from gate C terminates and returns the Test_Site shown at gate A. The trace from gate-F returns the output of gate-D as a candidate Test_Site for an XOR-type cell.

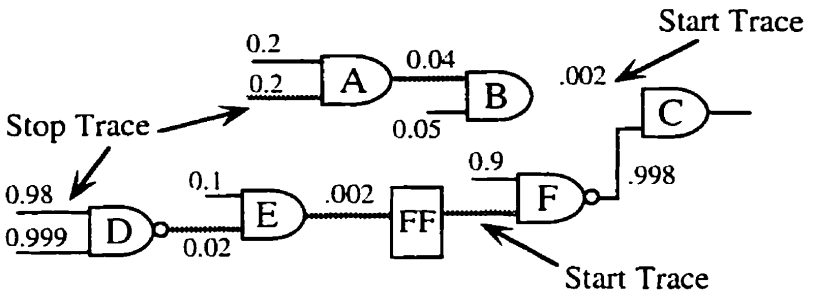


Figure 4.7: Example of Backtrace and Test_Site Extraction

The effect of a candidate controllability point is reflected in a temporary switching profile which is found by propagating the associated injected bias offset (table 4.2) through the circuit. The rank of a candidate is based on features of this predicted post-insertion switching profile compared to the previously existing switching. It is a weighted sum of the net number of fixed polarity nodes forced to switch, the net number of failing OL_i lines repaired and the net number of newly switching flip flops. If all candidates possess a rank less than or equal to zero, candidates are re-accumulated using a different set of trace thresholds or the rank function weights are altered.

In Figure 4.8, the controllability points are inserted for the Test_Sites found in the example of Figure 4.7. The mode-select gate is omitted. Updated biases are shown along the paths affected by each point and the repaired lines are shown dashed. If the candidate modifications are ranked according to the number of repaired OL_i values, both points are of equivalent value (they each repair 1 line). If the rank is weighted to favor an increase in the number of switching flip flops, Figure 4.8b is chosen (assume that a flip flop bias of $0.002 \approx 0.0$).

After each insertion, the reference switching profile is updated using either logic simulation or offset propagation. Since the estimates are prone to errors, simulation should be used periodically in order to align the data to exact (depending on simulation length) values. A final simulated switching profile, determined after all controllability points are inserted, characterizes the acceptable reference behavior of the CUT . All other circuit modifications should preserve this profile, or at least maintain the critical portion which indicates the number of zero switching flip flops and blocking lines.

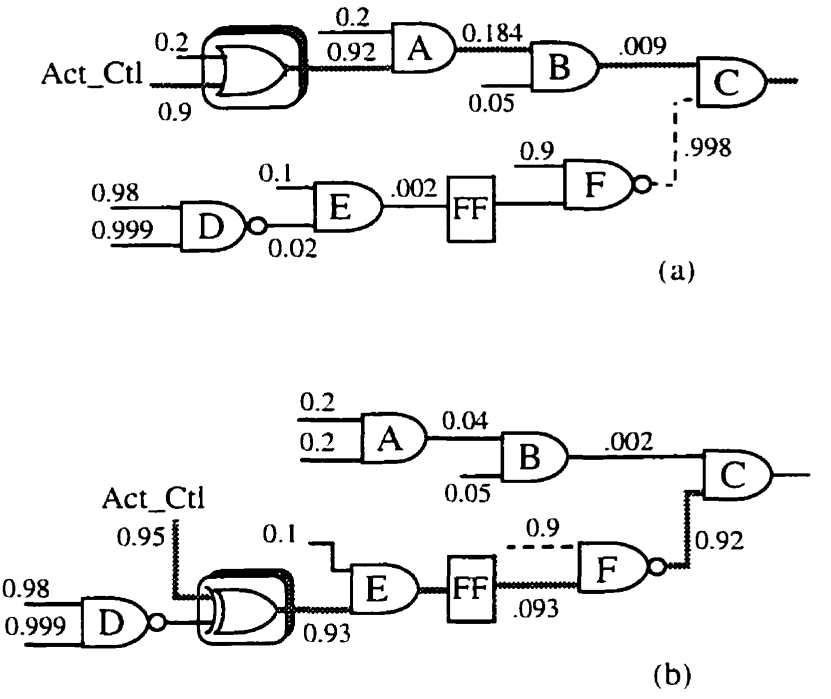


Figure 4.8: Effect of Controllability Point Insertion at Candidate Test_Sites - (a) OR insertion; (b) XOR insertion

4.3.2 Internally Driven Controllability Points

Each externally generated Act_Ctl signal is to be replaced with one tapped from existing circuit logic. Since previously established switching must be preserved, the key is to extract candidate *test source* lines from circuit regions suspected to be uncorrelated to the activity of the respective controllability point. This is done using a process of elimination which discards a candidate line if the pairwise fault free observability between it and the other lines which either influence or are affected by test point activity, exceeds a user-defined *prune threshold.* Relative to a controllability point, the removed areas are illustrated in Figure 4.9. These subcircuits relate to:

- the lines affected by the controllability point (the *activity front*)
- the backcone of each line in the activity front, including the test point itself
- the lines observable from stems pruned in backcone regions

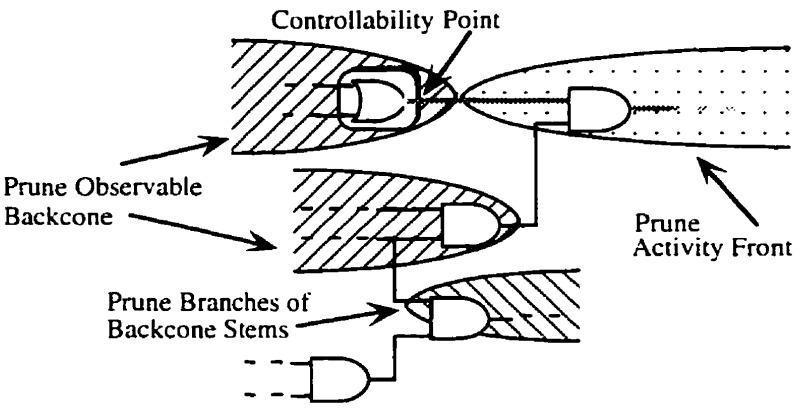


Figure 4.9: Removed Regions Suspected to be Correlated to a Controllability Point

The activity front can be mapped using logic simulation or via offset propagation where the injected offset disables the respective Act_Ctl signal. Four distinct sets of prune thresholds provide sufficient flexibility to regulate the filtering - one set for each of the above regions and an additional one specifically for removing the backcone of the controllability point itself. 'Sets' are used so that prune thresholds can be varied over consecutive time frames (recall from Section 4.2.1, observability is estimated over an arbitrary number of time intervals defined by the traced flip flops).

The observability time frame depth and prune thresholds can be manipulated depending on the circuit being tested. As test source candidates traced within a low sequential depth are most likely to distort the switching profile, the thresholds of the first two time frames are set to the minimal values, while those for deeper sequential distances may be increased.

Given the lines remaining after pruning, those which approximate the respective controllability point's Act_Ctl bias or inverted Act_Ctl bias are candidate test sources. In order to assist error propagation, candidate test sources are ranked so as to tap signals from lines possessing minimal OL_i. Similarly, a secondary ranking can be based on maximizing the number of low OL_i lines in the backcone of the test source. Of course, the final check uses simulation or offset propagation to ensure that the switching profile is maintained by the tentative connection. In the event that suitable candidates cannot be found, the prune thresholds and the time frame depth can be adjusted (i.e. increase the prune threshold and reduce the depth) to increase the candidate pool. When found, the selected test source line is split to also drive the controllability point and

the associated temporary external Act_Ctl signal is discarded. Connections which create a combinational loop are not permitted.

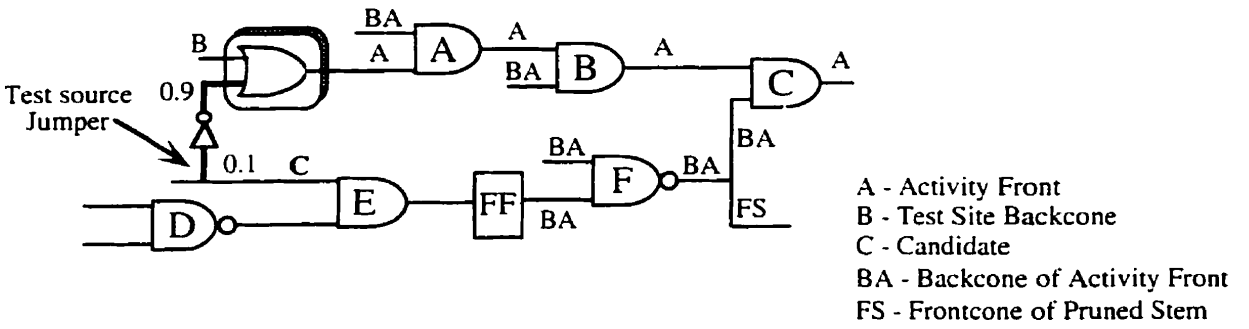


Figure 4.10: Selection of Internal Test Source

Figure 4.10 completes the previous test point insertion example of Figure 4.8a. All pruned lines are labeled relative to the controllability point. A time frame depth of zero is used for pruning and the associated observability prune threshold in each threshold set is assigned a value of 0.001. Thus, all lines aside from those input to the flip flop are eliminated. The line tagged C approximates the inverted Act_Ctl bias so it is used to source the controllability point.

```
1) Get reference switch profile with all externally driven controllability points
2) Select an untried controllability point & associated external source
3) Identify and prune lines correlated to the point
4) Collect & rank internal test source candidates
5) If an untried candidate test source exists
        5a) Connect untried test source to controllability cell
        5b) Predict temporary post-connection switch profile
        5c) If reference switch profile preserved
                5d) Update reference switch profile
                5d) If all external controllability points processed
                        5e) DONE (--> proceed to observation point insertion)
                else
                        5f) Go to (2)
            else
                5g) Go to (5)
  else
        5h) Postpone attempts for this replacement and goto(2),
         or revise prune thresholds and go to (4)
```

Figure 4.11: Procedure for Replacement of Externally Derived Controllability Signals

An iterative procedure for the replacement of externally derived controllability signals is summarized in Figure 4.11. To account for approximation errors and to properly update the reference switching profile, the circuit should be re-simulated after each external signal replacement (or at least periodically).

4.3.3 Results 1 - Switching Profile Adjustment & Fault Coverage

The two stage procedure of inserting self-driven controllability points is evaluated using eleven of the ISCAS-89 sequential benchmark circuits[Brg89b]. In each CUT, internal flip flops are synchronized to a single clock signal and initialized to the 0-state. The pseudorandom test length is arbitrarily set to 100K patterns.

When determining Test_Site locations, manipulation of the trace thresholds is anticipated (e.g. to vary the number of candidates). For the experiments performed, the block threshold ranges between 0.00001 to .005, and the stop threshold starts at 0.1 and decreases to 0.002. Xor-select is approximately 0.9 to 0.95. As a rule of thumb, the stop threshold should be chosen so that the component of the Test_site input to the controllability point remains sensitive with an OL_i value greater than the current block threshold. Extremely low stop thresholds were rarely needed.

The external Act_Ctl bias ranges are {0.1, 0.25, 0.5} for AND-type, {0.5, 0.75, 0.9} for OR-type and {0.05, 0.25, 0.5, 0.75, 0.95} for XOR-type controllability points. In the case of loops that converge to a fixed state, insertion of candidate XOR-type points with Act_Ctl bias of 5%, 50% or 95% are explored as an exit condition. Also, as an option, highly correlated gate inputs are tagged for splitting with an XOR-type test site (AND/OR-type could also be used.) Test_Mode is assigned a bias of 95%.

Externally driven controllability point candidates are ranked to favor an increase in the number of switching flip flops and, as a secondary measure, favor either a decrease in the number of fixed polarity lines or a reduction in the number of the zero OL_i lines. For the smaller circuits (s420, s444, s526n, s820, s832 and s838) simulation runs of 2000 to 5000 patterns are used to

update the switching profile after each insertion. The larger ones initially use simulation only periodically, and 2000 simulated patterns are used to update the profile for the last 30%-40% of the modifications. These latter simulation runs were needed since the circuits contained a high number of redundancies which affected the testability estimates. In all cases, computational cost is reduced by using Act_Ctl ranges narrowed to one member value and expanding the evaluation range if finding useful controllability points repeatedly fails near the end of the process. Although a degree of latency in switching can exist in some structures, for instance counters, justification of such short simulation lengths is based on the assumption that if a line fails to achieve the desired switching properties relatively quickly (within a few thousand patterns) it has either reached a near steady state signal probability or the chance of its detection within 100K trials is diminished. In addition, since the process is iterative, shortened simulation runs saves CPU effort.

Except in s9234 and s38417, for each controllability point replaced by a self-driven one, 3000 patterns are used to map the required activity front and update the post-insertion switching profile. The two largest circuits use offset propagation to perform the first task. This approximation reduces computational effort and provides acceptable results. In this phase, prune thresholds range from 0.001 to 0.05 and the maximum sequential depth is 10.

Table 4.4 outlines the number of non-switching and zero OL_i circuit lines for three representations of each benchmark. Each of these cases reflects a stage of the modification process - before DFT ("Original Ckt."), after insertion of externally driven controllability points ("External Ctl.") and after replacement of external test signals ("Self-Driven"). The number of non-switching flip flops and primary outputs are shown for the unmodified CUT but later omitted since none of these elements fail in the latter profiles.

For each benchmark, with a number of controllability points ("#Ctl Pts") amounting to a fraction of the number of flip flops (46% on average for circuits with 32 flip flips or fewer, and as low as 6% for the larger ones), an acceptable switching profile is achieved. Moreover, by inspection, apart from some minor variations, the local activity characterized in the externally driven case is sufficiently retained in the switching profile of the associated self-driven circuits.

Circuit Data			Original Ckt.					External Ctl.			Self-Driven		
			Fail Switch			Fail OL_i		Fail Switch	Fail OL_i		Fail Switch	Fail OL_i	
Name	#FF	#Ctl. Pts.	lines	FFs	POs	OL_0	OL_1	lines	OL_0	OL_1	lines	OL_0	OL_1
s420	16	7	98	12	1	140	155	0	0	0	0	0	0
s444	21	10	21	1	0	47	50	0	3	0	0	3	0
s526n	21	7	122	8	4	186	161	0	0	0	1	3	4
s641	19	1	22	4	1	28	46	0	0	0	0	0	0
s820	5	3	61	0	9	112	110	0	0	0	0	0	0
s832	5	5	67	0	9	119	127	0	0	0	0	0	0
s838	32	13	215	28	1	310	376	0	0	0	0	0	0
s1423	74	25	75	10	0	88	162	0	0	4	0	0	4
s5378	179	42	332	22	2	552	562	0	0	4	0	0	11
s9234	228	86	1165	109	2	1628	2361	0	18	12	5	30	44
s38417	1636	93	3303	630	4	5932	5026	8	80	40	14	75	45

Table 4.4: Controllability Point Insertion - Fault Free Switching Profiles for 5000 Random Patterns

Particularly for the larger circuits, because of the existence of redundancies and sequential untestable faults[Lia95], controllability point insertion is sometimes halted before an ideal all zeroed switching profile is reached. It is also suspected that fewer modifications would be needed if redundancies are absent. Nevertheless, these benchmarks are still useful in demonstrating the validity of driving the points with existing circuit lines.

Pertaining to each circuit representation described in table 4.4, table 4.5 contains the stuck-at fault coverage after ATPG and after fault simulating 100K pseudorandom test patterns. Note that given a benchmark, although the fault list size differs slightly in each of the three associated cases, the relative magnitude of the results is a valid criterion for comparison. The simulation-based test generation tool CRIS[Saa92] is used for all experiments.

The following analysis of table 4.5 demonstrates the effectiveness of self-driven controllability points. As implied by the failing switching profiles of table 4.4, pseudorandom fault coverages for the original circuits are unacceptably low. These range from 5.8% to 87.4% . In contrast, the corresponding fault coverages for the modified circuits are substantially higher. Considering the eleven self-driven representations, three attain 100% coverage and six exceed

93%. In the remaining three instances, s1423, s9234 and s38417, although the fault coverages are only 89.3%, 75.8% and 84.7% respectively, these values are marked gains relative to 59.4%, 5.8% and 13.1% respectively for the unmodified versions. In fact, the pseudorandom test results for the internally controlled test system is comparable to, if not better than, the externally controlled counterpart. These observations coupled with the preservation of the switching profile confirm that self-driven controllability hardware can be created for a number of circuits.

Sometimes, compared to the externally controlled system, the new connections of SD-BIST can increase the observability of some faults. The higher rate of pseudorandom detection for s9234 and s38417, and the higher ATPG coverage for s838, s1423, s9234 and s38417 suggest the this effect. In situations where pseudorandom fault coverage drops slightly, such as in s526n, the test length can be increased, a different set of internal connections can be attempted by manipulating modification thresholds or observability points can be inserted.

Circuit	#Ctl. Pts	ATPG Cov (%)			100K Patterns Cov (%)		
		Original Ckt	External Ctl.	Self-Driven	Original Ckt	External Ctl.	Self-Driven
s420	7	60.4	99.6	98.7	87.2	100	100
s444	10	86.9	96.2	96.2	17.5	94.9	97.4
s526n	7	71.2	96.7	94.7	13.2	94.4	93.8
s641	1	84.8	94.9	97.0	87.4	99.9	100
s820	3	45.9	94.2	79.7	51.5	98.3	97.7
s832	5	46.5	93.3	95.7	50.4	97.0	96.2
s838	13	36.8	83.2	98.8	49.1	100	100
s1423	25	87.4	87.9	93.5	59.4	88.8	89.3
s5378	42	68.6	86.3	85.5	69.1	94.4	95.1
s9234	86	5.8	37.9	66.5	5.8	47.6	75.8
s38417	93	13.1	19.4	50.1	13.1	80.9	84.7

Table 4.5: Fault Coverage - Controllability Point Insertion

4.4 Observability Enhancement

After self-driven controllability points have been inserted in a fault-independent manner, the burden of attaining high fault coverage is shifted to observability enhancement in the fault domain.

Observability test points are introduced into the circuit after acceptable line controllabilities have been established. Assuming the existence of an undetectable fault, an observability point transfers an error signal from an *error source* line affected by the fault to another internal *target* site unaffected by the fault and from which observation at a state variable or primary output is probable. As illustrated in Section 4.1.1, these connections are intrusive rather then simply probing in nature (e.g. in [Fox77]). Therefore, it is crucial that modifications inflict low area and delay penalties, maintain predictable fault free switching and not cause the target sites themselves to become undetectable. The cells of Figure 4.12 and the insertion procedure outlined in the rest of this section achieve these goals.

4.4.1 Observability Cells

The observability cells shown in Figure 4.12 differ in test mode multiplexing logic (gate M), driven by the error source (Err_Src) and the test mode flag (Test_Mode), and transfer logic (gate T) which gates the output of M and the signal at the target site (Target). The cell output is connected to the node which was originally fed by the target site.

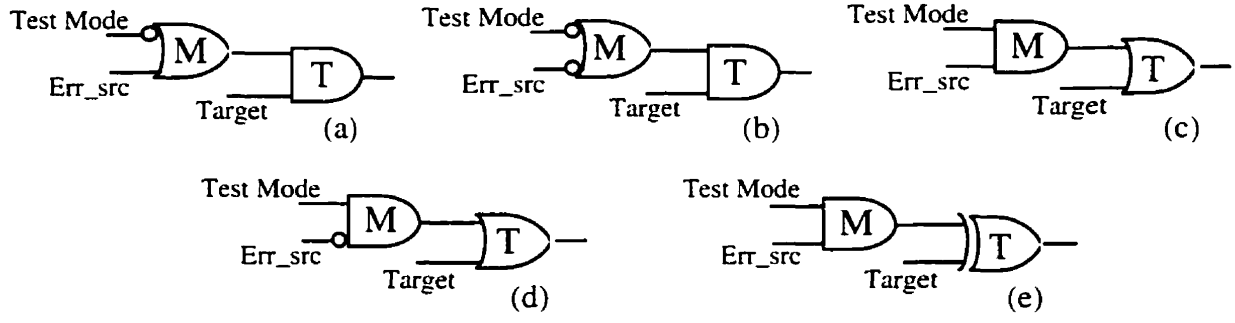


Figure 4.12: Observability Cells.

Enumerated in table 4.6, cell selection is guided by the fault free signal probabilities at the error source and target lines. Provided that the target line chosen is uncorrelated to the error source, these criteria are such that the degradation of OL_i at the transfer gate inputs is minimized. Likewise, since the extent to which the switching profile is altered is related to the difference between the target site bias and the output bias of the observability cell, these insertion rules also curb the deviation in circuit switching.

Err_Src Bias	Target Bias	Observability Cell (Fig 12)
» .5	≥ .5	a
« .5	≥ .5	b
« .5	< .5	c
» .5	< .5	d
free	≈ 0.5	e
≈ 0.5	free	e

Table 4.6 - Rules for Observability Cell Selection

For cells of Figure 4.12a-d, sensitivity and switching maintenance properties progressively improve as the error source bias approaches 1.0 or 0.0. XOR-type cells may be used if either the target or error source bias is nearer to 0.5. Note, XOR-type cells can sometimes be interchanged with the others.

Placement is regulated by fault free global observability to the state variables and primary outputs, error probability in the presence of an injected fault and user-defined thresholds. Figure 4.13 is a summary of the suggested procedure.

1) Generate switching profile & fault simulate to get undetected fault list
2) Map error propagation for each fault & collect error sources
3) Get candidate error source-target pair by pruning regions correlated to faults covered by the error source
4) If candidate connections exist
   4a) For each error source :
      4b) Rank the candidate error source-target connections
      4c) If connections with acceptable rank exist
            4d) Insert observability cell connection with highest rank.
         else
            4e) Defer or adjust thresholds & Go to (2)
   else
      4f) Terminate, or adjust thresholds & Go to (2)
5) Logic simulate to revise switching profile & fault free observability measures
6) Drop faults (a) covered by error source used or (b) suspected undetectable due to inactivity & Go to (2). At this step, since the method is not exact periodically Go to (1)

Figure 4.13: Overview Procedure for Observability Point Insertion

The initial preparation step is the compilation of a reference fault free switching profile and the extraction of a list of undetectable faults composed of those faults not detected after fault simulating a number of test patterns which approximates, but does not exceed, the expected test length[2]. Note that this means of classification offers no distinction among untestable faults and testable faults not covered at the end of the fault simulation run. The two major tasks which remain concern the determination and rank of candidate error source-target connections.

4.4.2 Determination of Candidate Error Source-Target Pairs

The error source line selection problem is twofold and similar to probe point identification as done in [Fox77][Iye89][Tou96]. First, for each undetected fault, fault simulation tracks the circuit nodes to which fault effects propagate. Offset propagation can also be used to inject a fault effect but fault simulation is much more accurate. Next, a greedy approximation to a set covering problem[3] chooses a small number of possible error source lines which possess a

[2] A test length of 100K pseudorandom patterns is again used for experimentation.
[3] The exact problem is NP-complete.

threshold error probability for each fault mapped to the respective source line. If circuit modifications are limited to a subset of the proposed observability cells, an additional filtering criterion is required. For instance, if cells are limited to among Figure 4.12a-d, permitted error source lines are those with a fault free bias within a threshold of either 1.0 or 0.0

For each candidate error source, possible target sites are those lines which are uncorrelated to all faults covered by the source. These target candidates are gathered using a pruning method similar to the one in Section 4.3.2 except that the fault free activity front is replaced by the combined region of error propagation due to all faults associated with the respective error source. In addition, if at least one input to a gate is reached by an error, all other inputs to the gate and their associated backcones are discarded. Only target sites with a non-zero fault free observability to the state elements or POs need be retained.

4.4.3 Rank & Selection of Observability Points

The rank of an observability cell candidate reflects the probability that a transferred error will, in the same cycle, reach either the state elements (State_rank) or the primary outputs (PO_rank).

Given an error source-target pair, in accordance to the observability cell structure dictated by table 4.6 , the error probability, ε_{out} , at the output terminal is estimated by gating the error probability at the Err_Src line, ε_{in} , and the other signal probabilities input to the cell. As observability cell inputs are chosen to be uncorrelated, Equations 4.5 to 4.7 approximate ε_{out} for the circuits of Figure 4.12:

AND-transfer logic (fig. 12 a,b)

$$\varepsilon_{out} = \varepsilon_{in} \times P_{Target} \times P_{Test\_Mode} \qquad \text{Equation (4.5)}$$

OR-transfer logic (fig. 12 c,d)

$$\varepsilon_{out} = \varepsilon_{in} \times (1-P_{Target}) \times P_{Test\_Mode} \qquad \text{Equation (4.6)}$$

XOR-transfer logic (fig. 12 c,d)

$$\varepsilon_{out} = \varepsilon_{in} \times P_{Test\_Mode} \qquad \text{Equation (4.7)}$$

where,

ε_{in} :	Minimum error probability do to all faults covered by the error source
P_{Target}:	Fault free signal probability at the Target terminal
P_{Test_Mode}:	Fault free signal probability at the Test_Mode terminal

The rank is the product of ε_{out} and the combinational (i.e. depth 0) fault free global observability from the target site to the state elements or POs. The required observabilities are calculated along paths unaffected by the faulty activity and in the absence of the candidate cell. Furthermore, as shown in Figure 4.14, only the observability of the non-controlling bit polarity at the target site is applicable, otherwise, error propagation is blocked within the cell. This approach avoids repeated recalculation of fault-domain testability measures and is possible partially because the target site is purposely chosen to be unaffected by the error source.

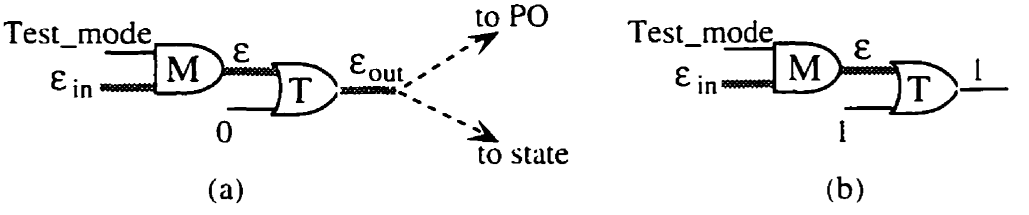


Figure 4.14: Error Propagation within Observability Cell - only the observability of a non-controlling '0' applies.

State_rank and PO_rank are calculated using Equations 6 to 9 :

AND-transfer logic (fig. 12 a,b)

$$\text{State\_rank} = \varepsilon_{out} \times \text{target\_stateobs1} \qquad \text{Equation (4.8)}$$

$$\text{PO\_rank} = \varepsilon_{out} \times \text{target\_POobs1} \qquad \text{Equation (4.9)}$$

OR-transfer logic (fig. 12 c,d)

$$\text{State\_rank} = \varepsilon_{out} \times \text{target\_stateobs0} \qquad \text{Equation (4.10)}$$

$$\text{PO\_rank} = \varepsilon_{out} \times \text{target\_POobs0} \qquad \text{Equation (4.11)}$$

XOR-transfer logic (fig. 12e)

$$State\_rank = \varepsilon_{out} \times (target\_stateobs0 \times (1\text{-}P_{Target}) + target\_stateobs1 \times P_{Target}) \quad \text{Equation (4.12)}$$

$$PO\_rank = \varepsilon_{out} \times (target\_POobs0 \times (1\text{-}P_{Target}) + target\_POobs1 \times P_{Target}) \quad \text{Equation (4.13)}$$

where,

target_stateobs0/1: Combinational fault free observability of a 0/1 (the non-controlling bit value) at the target line to the state elements *before* current cell insertion.

target_POobs0/1: Combinational fault free observability of 0/1 at the target line to the primary outputs *before* current cell insertion.

Equations 4.8-4.13 can be altered to include an increase in rank if the error frontier mapped before observability cell insertion already extends over sate elements or primary outputs (e.g. a real valued ORing of new and existing propagation effects). This is not shown here since experiments suggest that the effect of such a modification can usually be omitted.

The selection of a candidate observability cell is based on a weighted sum of PO_rank and State_rank. If both terms are zero (or below a threshold), the insertion is either postponed or net increase in the number of lines reached by the transferred error is the deciding factor. Here, the deferral option is used. If a suitable connection is found, the faults covered by the respective error source are dropped from future processing. Fault simulation can be performed periodically to accurately revise the undetected fault list. Modifications which introduce combinational loops are not permitted.

Observability point insertion continues until either the fault coverage is considered acceptable or a pre-determined number of insertions is reached. If premature termination occurs due to the failure to find error source-target pairs, the procedure is restarted using alternate thresholds. Typically, if insufficient target sites exist then the pruning thresholds are reduced. The pool of error sources is increased by reducing the threshold error probability and/or relaxing threshold bias requirements (section 4.4.2). Experimental values for these guides are presented below.

4.4.4 Results 2 - The Self-Driven System

Experimental results for observability point insertion are presented for the circuits of Section 4.3.3. In each case, apart from insignificant deviations, the critical portion of the reference "Self-Driven" switching profile found in table 4.4 remains unchanged (all zeroed entries remained as such). Given a list of faults which are yet undetected after the controllability enhancement phase, fault simulation runs ranging between 200 to 5000 patterns are used to determine the set of candidate error source lines. In order to reduce execution time, for larger circuits, a maximum of 500 patterns are used in this context. Also when dealing with larger circuits, in the initial iterations if the scheme, the set of undetectable faults to be processed is reduced by discarding those faults reachable from another with at least a fault free observability threshold of 3%.

For experiments performed, only AND/OR-type cells are used. The threshold error probability at the error source is approximately 0.0005 and bias at that site is limited to within 0.1 of 0.0 or 1.0. The prune thresholds are set to 0.005 for backcone regions and 0.2 for stem regions (section 4.3.2). A time frame depth equal to 2 is used for observability calculations during pruning. As mentioned, these thresholds can vary during the process.

Observability point insertion favors the candidate pair with a PO_rank exceeding 0.001 and possessing the maximum State_rank, otherwise, the connection with the highest state rank over 0.0001 is chosen. These limits can be adjusted but should imply the detection of the fault at the PO within 100K.

ATPG and pseudorandom fault coverage for the improved benchmarks are shown in table 4.7. Since the pseudorandom pattern coverage surpasses that of ATPG, observability point insertion for the larger circuits stops when the incremental increase in detection for additional points is between 2 to 5 additional faults for a large fault set.

Circuit	#FF	#Ctl. Pts.	#Obs. Pts.	ATPG		100K Patterns			% Area Overhead
				% Cov	#undet	% Cov	#undet	% Gain	
s420	16	7	0	98.7	7	**100**	0	-	5.1
s444	21	10	0	96.2	23	**97.4**	17	-	8.0
s641	19	1	0	97.0	16	**100**	0	-	0.3
s838	32	13	0	98.8	12	**100**	0	-	3.0
s526n	21	7	0	94.7	39	93.8	63	-	4.6
			1	94.5	41	94.0	45	28.6	5.0
			5	95.9	31	**97.0**	26	58.7	6.8
s820	5	3	0	73.6	288	97.7	25	-	2.0
			1	88.1	131	99.7	3	88.0	2.4
			2	85.2	164	**100**	0	100	2.8
s832	5	5	0	93.9	70	96.2	43	-	3.2
			3	96.6	40	99.1	11	74.4	4.5
			12	95.5	55	**100**	0	100	8.0
s1423	74	25	0	93.5	113	89.3	188	-	5.0
			8	93.3	121	95.5	81	56.9	6.0
			17	94.9	94	**97.8**	40	78.7	7.2
s5378	179	42	0	85.5	782	95.1	300	-	2.7
			7	86.4	736	96.7	180	40.0	3.0
			26	91.0	497	**97.7**	128	57.3	4.0
s9234	228	84	0	66.5	2696	75.8	1950	-	2.8
			12	83.4	1353	82.4	1602	17.8	3.0
			22	78.7	1743	**84.0**	1350	31.0	3.2
s38417	1636	93	0	50.1	17266	84.7	5294	-	0.70
			9	55.3	14870	85.0	4979	6.0	0.74
			22	62.8	12418	90.0	3270	38.2	0.80
			28	61.0	13019	**92.2**	2425	51.2	0.85

Table 4.7: Fault Coverage and Area Penalty - Self-Driven Controllability & Observability Points

Given the circuits modified for controllability, aside from s444, observability connections succeed in reducing the number of yet undetected faults ("#undet"). For pseudorandom testing, a measure of this is shown in the "% Gain" column. Here, for large initial undetectable

fault lists, even small gains can be substantial. For example, in s38417 a 6% gain is equivalent to 315 newly detected faults with just 9 observability points.

Of the four circuits that show a pseudorandom fault coverage below 95% before observability modifications, s526n and s1423 now exceed 97%. As in Section 4.3, the lower fault coverages of s9234 and s38417 are probably due to a high number of redundancies, however, the results are a substantial rise from the initial values of 5.8% and 13.1% respectively for the unmodified circuit.

Using an industrial standard cell library, the area overhead due to test point insertion is estimated and presented in table 4.7. These values do not include the contribution of wiring. Compared to the unmodified CUT, the overhead of the self-driven system ("% Area Overhead") is acceptably small - on average 4.5% for the CUT versions containing the most test points, and approximately 1% for the largest CUT. In fact, the apparent trend is that this area measure decreases with increasing CUT size. Since the XOR-type controllability point is the biggest cell, increasing the xor-select criteria so that fewer XOR-type cells are inserted will reduce the hardware cost. The removal of redundancies from within the CUT is expected to have a similar effect.

Table 4.8 compares the performance of SD-BIST with two existing non-scan test-per-clock schemes. The pseudorandom test length in all cases is 32K patterns and although fault set redundancy and composition differ due to test hardware and methodology, the absolute fault coverage ("% Cov") is interpreted as an evaluation criterion. The other ranking factor is a lumped site count ("#sites") which, as appropriate, refers to either the number of test points used or the number of flip flops modified.

Each listed system aims to establish flip flop switching. However, in a broader scoap, SD-BIST also considers sensitivity and switching at all other lines, and as it turns out, in most test cases SD-BIST succeeds in doing so with the lowest logic overhead and the highest fault

coverages. In the few instances where SD-BIST exhibits lower fault coverages, the opposing scheme usually requires significantly higher logic overhead.

Circuit	Partial Reset [Sou95a]		XFF [Sou95b]		XFF (+ obs. pts.)		SD-BIST Ctl. only		SD-BIST Ctl. + Obs.	
	#sites	% Cov	#sites	% Cov	#sites	% Cov	#sites	% Cov	#sites	% Cov
s420	15	85.2	16	88.5	>16	95.0	7	100	-	-
s444	14	91.6	21	97.4	-	-	10	97.4	-	-
s526n	14	82.4	21	100	-	-	7	91.1	12	94.1
s641	4	98.7	19	98.0	-	-	1	100	-	-
s820	0	50.5	5	99.9	-	-	3	96.2	5	99.3
s832	0	49.7	5	98.3	-	-	5	95.3	12	99.6
s838	30	69.8	32	58.1	>32	86.6	13	100	-	-
s1423	22	67.4	74	97.3	-	-	25	87.0	42	96.5
s5378	94	90.0	179	94.9	-	-	42	93.5	68	97.1
s9234	30	28.8	228	92.2	-	-	86	64.1	106	80.6
s38417	70	43.6	1635	69.8	>1635	94.8	93	76.4	121	88.6

Table 4.8 : Comparison of At-Speed DFT using Partial Reset [Sou95a], XFFs [Sou95b] and SD-BIST ; Pseudorandom test length = 32K

Relative to the partial reset technique of [Sou95a], a fair comparison considers the SD-BIST system without observability enhancement. Even so, it is found that SD-BIST provides superior fault coverage with usually fewer inserted cells. It can be also inferred that the restrictive nature of the partial reset modifications limits the number of altered sites thus resulting in the lower fault coverages. This is especially evident in the low site count and comparatively low fault coverage of s820, s832, s838, s9234 and s38417, In fact, no resettable cells could be placed for s820 and s832. However, as claimed in [Sou95a], the partial reset results may be improved if test point insertion is combined into the system. Indeed, this suggestion indicates one of the strengths of SD-BIST - it can be used to augment and improve the efficiency of another DFT approach without requiring additional external control access.

As noted in chapter 3, since flip flops are parallel loaded via primary input connections, the non-scan pseudorandom method of [Sou95b] retains the direct state access of scan. In addition, as the observability points are placed at flip flop outputs, the full state observability aspect of scan is also approached. These properties can also cause the redundancy of the fault set in [Sou95b] to differ significantly from that of SD-BIST. Nevertheless, for the sake of completeness and since the test application format is simliar to that of SD-BIST, results for [Sou95b] are also listed in table 4.8. Like SD-BIST results, the data is split into two sets, with and without observability cells. Unfortunately, in [Sou95b], there is no indication concerning the actual number of observability points used. Despite this, comparing the systems which use only controllability modifications, in 6 cases (s420, s444, s641, s838, s5378 and s38417) SD-BIST provides comparable or better absolute fault coverage with fewer inserted cells. The associated logic saving is markedly significant in s5378 and s38417. If the observability connections of SD-BIST are considered, s1423 and s820 can be added to the list of circuits for which XFF is outperformed. The remaining cases are conservatively judged as comparable due to the differences in redundancy and because additional test points can be added (e.g. up to the number of FFs) to improve the fault coverage of SD-BIST. As above, self-driven test points can be seamlessly integrated into the method of [Sou95b].

Thus, by the above discussion, it can be concluded that in terms of the number of test points required, fault coverage and adaptable implementation style (due to reduced dependency on external signals), SD-BIST is a competitive stand-alone DFT scheme which can also be used in conjunction with other DFT approaches.

4.5 Tradeoffs & Comments

Experimentation demonstrates that SD-BIST can be used to achieve high stuck-at fault coverage using parallel pseudorandom test patterns. As opposed to other test point schemes SD-BIST does not require scan for test pattern application or test point control, and so is not penalized by the overheads usually associated with that platform. Here, the major source of delay in all test

cells is due to a single two input gate (an XOR in the worst case) at the test site/target site. The rest of the test hardware is isolated from normal circuit paths.

Apart from the logic overhead outlined previously, additional area is also required for reset circuitry, the pattern generator, the signature analyzer, the global test mode flag and internal test point connections. The reset logic can be eliminated if the CUT is set to an arbitrary known state using an initializing sequence. The particular starting state seems of little importance since once it is provided to the software, the controllability points will adjust the switching to a suitable form. As the test mode signal, test pattern generator and signature analyzer are required in typical BIST schemes, their contribution is considered a cost standard to most if not all competitive approaches. Layout level analysis is needed to properly evaluate the size and difficulty in routing the internal connections required for the self-driven format. In the worst conceiveable case, for each inserted cell the required connection corresponds to a location on the circuit boarder furthest from the cell. This is similar in effect to the global point-to-point wiring between chip boundaries and test points/flip flops of [Abr93] [Mat93] [Lin93] [Chi93] [Sou95a] [Sou95b] [Tou96]. Here, however, routing can be influenced by reducing the sequential depth and increasing observability thresholds during the circuit pruning phase. Relaxing the Act_Ctl value at a controllability point input can also influence pruning. In instances where a suitably close uncorrelated source/target cannot be found, additional control points can be added to logically partition the circuit during test mode. By doing so, regions forcably isolated from the modification site can be provided.

For controllability points (figure 4.5), it is possible to reduce the number of mode-select gates if, as in [Iye89] some controllability cells share a common Act_Ctl source. One extension that would enable this feature is to weight the rank of the internal source candidates (section 4.3.2) to favor previously used Act_Ctl sources. The speed degradation due to observability modifications can be decreased if instead of exclusively using the observability cells of Figure 4.12, the implementation scheme is adapted to rerouting the outputs of a number of space compaction circuits [Fox77] [Rud92]. In this way, each condensation block merges a number of observability point outputs. The tradeoffs of such hybrid schemes also involve the size and

relative ease building condensation blocks versus that of using a number of independent observability cells.

The SD-BIST modification procedure is dominated by graph search techniques used to accomplish path tracing, and offset propagation used to evaluate candidate alterations. Given a circuit containing n gates, a single test point insertion can be done in $O(n^2 logn)$ operations. Note that for a combinational circuit, offset propagation (floating point operations) can be done in linear time, however, for a sequential circuit, the procedure can be repetitive if sequential loops are traversed. Since it can be concluded that an injected offset which cycles through many states significantly affects circuit activity, such iterations are prematurely terminated (e.g. less than 20 iterations). Thus, in this scheme, the contribution of offset propagation remains relatively linear for sequential circuits.

Because the prototype software is constructed for flexibility rather than speed, a meaningful measure of the computation cost is offered in terms of the generic operations which can dominate the process. For controllability point insertion, repeated logic simulation is the most time consuming factor. In the worst case, logic simulation is performed once after each external test point insertion, once to map the activity front of each external test point and once after connection of an internal Act_Ctl source. For observability alterations, fault simulation to map the propagation of a fault and periodic revision of the fault list comprise the major effort. Thus, feasibility of this method can depend on the continued affordability of fault simulation. To offset this cost, testability estimation can be substituted for a number of faulty/fault free simulation runs.

Chapter 5 Conclusion

This thesis proposed a simple and effective means of using parallel pseudorandomly generated patterns to perform at-speed tests on non-scan sequential circuits. The method does not conform to the familiar solution format of providing access to state elements. Instead, the technique is based on the more general problem of strategically inserting a number of test points, therefore, any circuit line is a candidate for modification. Furthermore, as a departure from all existing DFT techniques, apart from the test mode flag, all control signals required for test point operation are tapped from within the CUT itself. Thus, logic such as scan and additional pins is not needed to introduce externally derived control signals into the test network. Moreover, as opposed to restricted, sometimes equiprobable, control biases used in conventional test point schemes, the new architecture presents a high degree of freedom in selecting these signal probabilities. Observability point outputs are connected to other internal nodes which are unaffected by the error being transferred via the point. The implementation also permits a reduction in number of independent observability connections by allowing the use of multiple test point condensing networks. In such cases, the new connection scheme is used to route the condenser block outputs to other internal circuit lines.

Logic simulation, proposed testability measures, path tracing and user-defined thresholds guide the test point placement. The scheme conducts much of the circuit analysis in the fault free domain. For instance, as the goal of controllability point insertion is simply to produce a threshold level of switching and line sensitivity, it is completely done using fault free analysis. This is different from other approaches which sometimes mix controllability manipulation with

fault oriented error propagation. In the outlined SD-BIST, fault domain procedures are only needed for observability point placement. Fault injection is required to determine internal probe sites, however, if the target line to which errors are transferred for observability is chosen as one unaffected by the errors, fault free observability measurements can be used to rank the value of the test point. Of course, this procedure can degrade to the use of fault domain observability measures for the same purpose but the fault free option is promoted as a computational saving.

It is inferred from experimental results that although the testability estimation procedure is prone to errors, their use as a guide to compare the relative effectiveness of modification alternatives is valid. Moreover, along with the relatively short test lengths found experimentally, the elimination of the need to serialize pseudorandom test data significantly reduces test time over scan based approaches. Also, although SD-BIST is presented as a stand-alone test scheme, it is adaptable for use in conjunction with any other DFT method which requires additional test points for improved fault coverage. The reduced dependence on externally derived control signals is uniquely advantageous to this end. On a similar note, instead using of the suggested test point insertion algorithms, the particular mechanics of test point placement can be solved using an available "externally-driven" scheme and then converted to a self-driven implementation.

An important characteristic of the new test system is the ability to adapt circuit switching given an arbitrary input signal probability distribution. Therefore, since the signal probabilities at the boundaries of embedded circuits may be far from what is suitable for test stimulation of subsequent logic, future work could examine the applicability of introduced techniques to testing cascaded or interacting sequential circuits. The flexibility of test-mode switching adjustment may also prove useful when considering low power designs. Other areas of investigation include experimentation with irredundant circuits, and reduction of test point hardware via sharing of controllability point inputs and introduction of condensation blocks.

Appendix A

This section lists equations which can be used in offset propagation. The approximation used can vary depending with the amount of sampled data available. It is assumed that offset injection occurs at a single site only.

AND Gates:

Assuming that input pin probabilities P_1 and P_2 are independent, the signal probability at an AND output is:

$$P_{original} = P_1 P_2 \qquad \text{Equation (A.1)}$$

Then, given an injected offset somewhere in the system,

$$P_{disturbed} = (P_1 + \Delta_1)(P_2 + \Delta_2) = P_1 P_2 + P_1 \Delta_2 + P_2 \Delta_1 + \Delta_1 \Delta_2 \omega \qquad \text{Equation (A.2)}$$

$$\Delta_{out} = P_{original} - P_{disturbed} = P_1 \Delta_2 + P_2 \Delta_1 + \Delta_1 \Delta_2 \omega \qquad \text{Equation (A.3)}$$

If correlation between Δ_1 and Δ_2 is neglected, w is set to 1, otherwise the formulae depends on the sign of Δ and the sampled probability of a particular bit pair. Additional error correction factors can also be introduced if sampled data is available to encapsulate correlation information at the input terminals of a gate. A compensation term is related to the ratio of the sampled and

expected probabilities of the bit pair affected by an input Δ. The bit pair in question depends on the sign of the bias offset and the gate type affected. For example, let S[i,j] be the sampled probability the bit pair [i,j] occurs on the first and second inputs to a gate. Then, given an AND gate and positive Δ_2, the affected pair is [1,0] (Δ_2 will force a new bit pair of [1,1] to replace [1,0]), and the new formula for bias offset propagation is :

$$\Delta_{out} = P_1\Delta_2\frac{S[1,0]}{P_1(1-P_2)} \qquad \text{Equation (A.4)}$$

If Δ_1 and Δ_2 exist:

Δ_1	Δ_2	Δ_{out}
>0	>0	$\Delta_1\frac{S[0,1]}{(1-P_1)}+\Delta_2\frac{S[1,0]}{(1-P_2)}+\Delta_1\Delta_2\omega$
<0	<0	$\Delta_1\frac{S[1,1]}{P_2}+\Delta_2\frac{S[1,1]}{P_1}+\Delta_1\Delta_2\omega$
<0	>0	$\Delta_1\frac{S[1,1]}{P_1}+\Delta_2\frac{S[1,0]}{(1-P_2)}+\Delta_1\Delta_2\omega$
>0	<0	$\Delta_1\frac{S[0,1]}{(1-P_1)}+\Delta_2\frac{S[1,1]}{P_2}+\Delta_1\Delta_2\omega$

Table A.1: Offset Propagation Formulae for 2-input AND.

Similarly, ω can be expressed as :

Δ_1	Δ_2	ω
>0	>0	$\frac{S[0,0]}{(1-P_1)(1-P_2)}$
<0	<0	$\frac{S[1,1]}{P_1P_2}$
<0	>0	$\frac{S[1,0]}{P_1(1-P_2)}$
>0	<0	$\frac{S[0,1]}{P_2(1-P_1)}$

Table A.2 - Correction Factors for $\Delta_1\Delta_2$ Term

It is found experimentally that the basic distribution formulae without error compensation are usually sufficient. Correction factors are an aid only when considering circuits with a high number of correlated lines or redundancies.

Note that the above equations can be expressed in terms of OL_i. This can be useful if testability estimation replaces consecutive simulation calls (in these cases simulation is used only periodically, thus computational effort is reduced). Rewriting OL_i for pin j of the 2-input AND as OL_i^j, the required relationships are as follows:

$S[0,1] \rightarrow OL_0^1$

$S[1,0] \rightarrow OL_0^2$

$S[1,1] \rightarrow OL_1^1 , \quad OL_1^2$

$S[0,0] \rightarrow 1 - OL_0^1 - OL_0^2 - OL_1^1$

OR Gates:

Offset Propagation:

$$P_{original} = P_1 + P_2 - P_1 P_2 \qquad \text{Equation (A.5)}$$

$$P_{disturbed} = P_1 + \Delta_1 + P_2 + \Delta_2 - (P_1 + \Delta_1)(P_2 + \Delta_2)$$

$$= P_1 + P_2 - P_1 P_2 - P_1 \Delta_2 - P_2 \Delta_1 - \Delta_1 \Delta_2 \omega + \Delta_2 + \Delta_1 \qquad \text{Equation (A.6)}$$

$$\Delta_{out} = P_{original} - P_{disturbed}$$

$$= \Delta_2 + \Delta_1 - P_1 \Delta_2 - P_2 \Delta_1 - \Delta_1 \Delta_2 \omega \qquad \text{Equation (A.7)}$$

Again, if correlation between reconverging offsets is ignored then ω is set to 1. Otherwise, they can be set as in table A.2. If correlation at gate inputs is considered, offset propagation formulae can be written as :

Δ_1	Δ_2	Δ_{out}
>0	>0	$\Delta_1 + \Delta_2 - P_2\Delta_1 \frac{S[0,0]}{(1-P_1)(1-P_2)} - P_1\Delta_2 \frac{S[0,0]}{(1-P_1)(1-P_2)} - \Delta_1\Delta_2\omega$
<0	<0	$\Delta_1 + \Delta_2 - P_2\Delta_1 \frac{S[1,0]}{P_1(1-P_2)} - P_1\Delta_2 \frac{S[0,1]}{P_2(1-P_1)} - \Delta_1\Delta_2\omega$
<0	>0	$\Delta_1 + \Delta_2 - P_2\Delta_1 \frac{S[1,0]}{P_1(1-P_2)} - P_1\Delta_2 \frac{S[0,0]}{(1-P_2)(1-P_2)} - \Delta_1\Delta_2\omega$
>0	<0	$\Delta_1 - \Delta_2 - P_2\Delta_1 \frac{S[0,0]}{(1-P_1)(1-P_2)} - P_1\Delta_2 \frac{S[0,1]}{(1-P_1)P_2} - \Delta_1\Delta_2\omega$

Table A.2: Offset Propagation Formulae for 2-input OR

The relations needed to express the equations of table A.3 in terms of OL_i^j are :

$S[0,1] \rightarrow OL_1^2$

$S[1,0] \rightarrow OL_1^1$

$S[1,1] \rightarrow 1 - OL_0^1 - OL_1^1 - OL_1^2$

$S[0,0] \rightarrow OL_0^1 , \quad OL_0^2$

NAND, NOR & Inverters

The Δ_{out} for an inverter is negative of the incoming offset. Similarly, offset propagation formulae for a NAND or a NOR is simply the negative of that used for an AND or an OR respectively.

Appendix B

This section compares components of the switching profile for the unmodified circuit ("Unmodified"), the intermediate circuit containing externally driven controllability points (Ext. Ctl), and the final circuit with self-driven controllability and observability points (SD-BIST).

For each benchmark, the four distributions presented concern the following parameters respectively: signal probability at each line, signal probability at the flip flops, OL_1 and OL_0. Recall that the critical portion of the profile which should be retained in the externally controlled and self-driven cases is the section corresponding to: near 0% or near 100% signal probability, and near 0% OL_1 and OL_0. Some deviation can be permitted in other regions without severely affecting the final fault coverages. As seen in the fault coverages of Chapter 4 and from the figures here, the SD-BIST case sufficiently tracks the Ext. Ctl case.

Note that due to test circuitry, the number of lines in the CUT increases slightly from the Unmodified to the final SD-BIST representation.

Circuit - s420

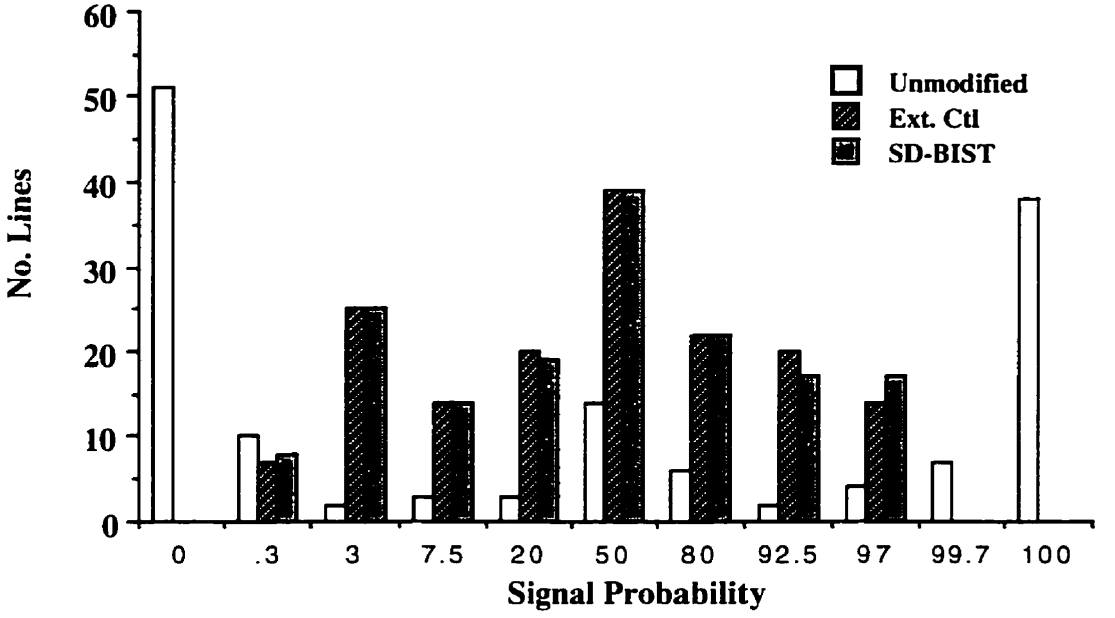


Figure B.1: s420 - Distribution of Line Biases

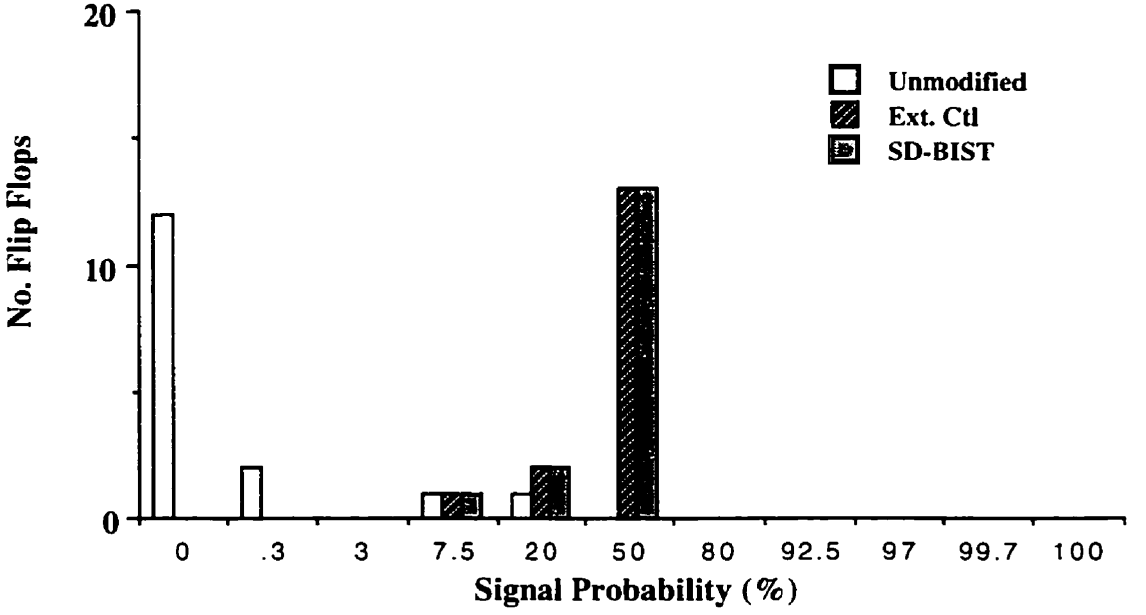


Figure B.2: s420 - Distribution of Flip Flop Biases

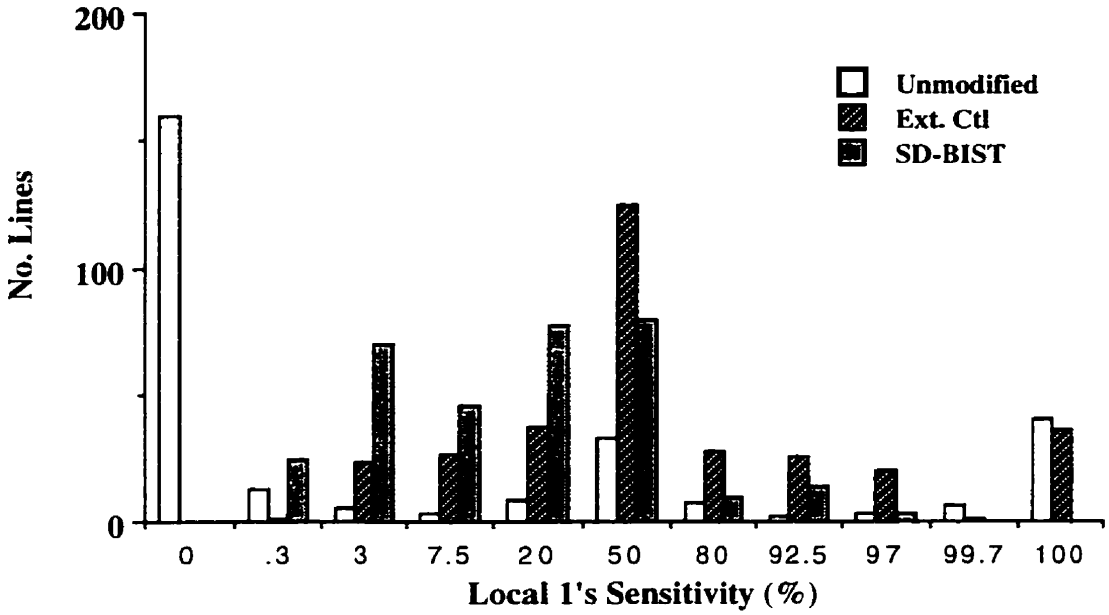


Figure B.3: s420 - OL_1 Distribution

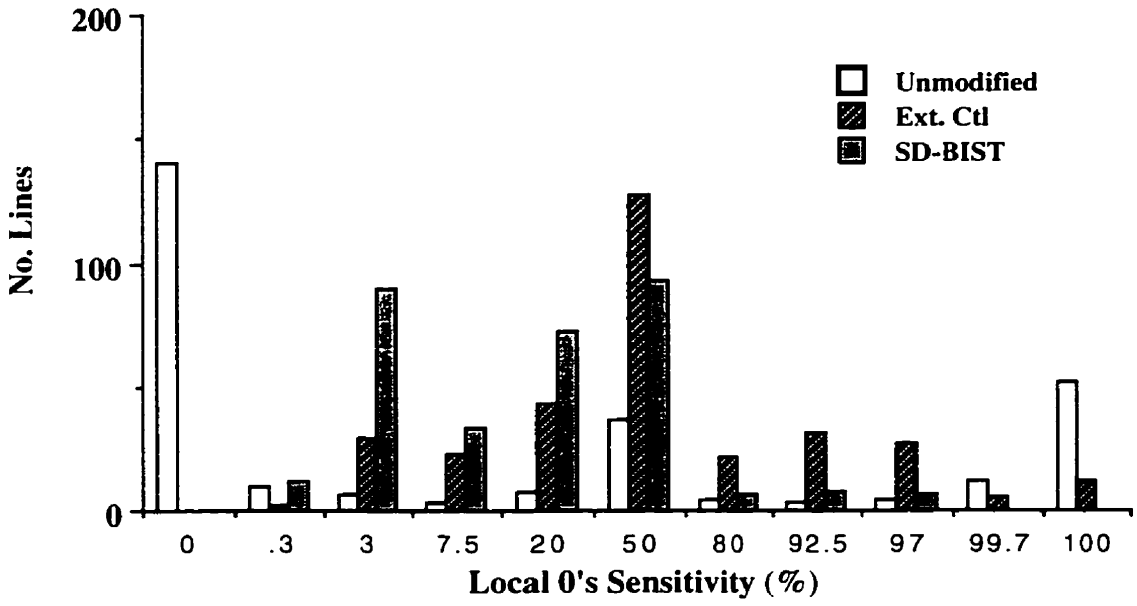


Figure B.4: s420 - OL_0 Distribution

Circuit - s444

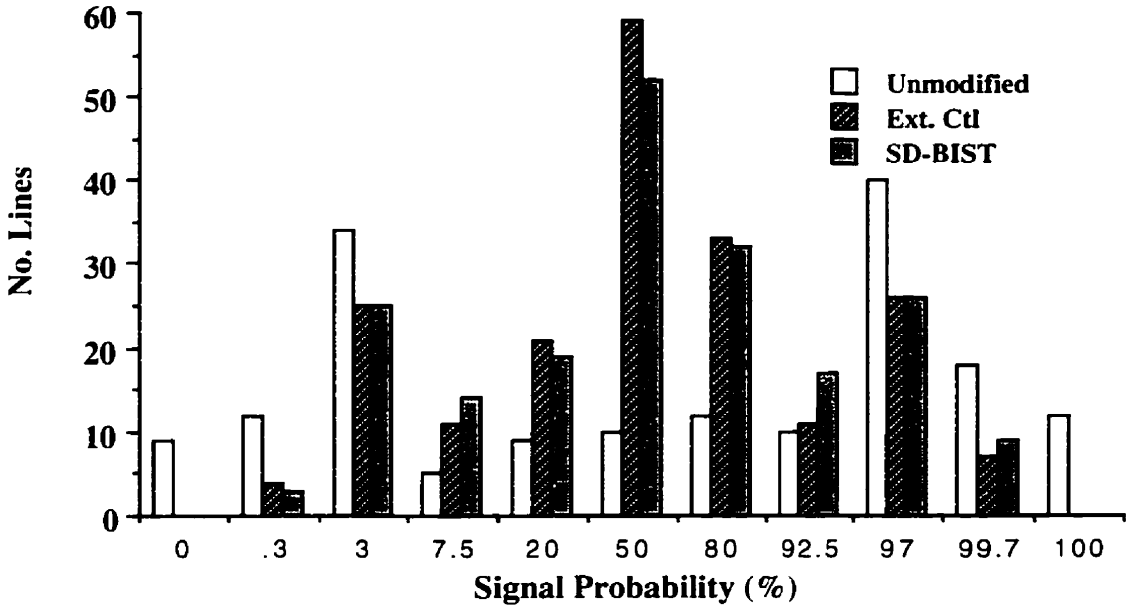


Figure B.5: s444 - Distribution of Line Biases

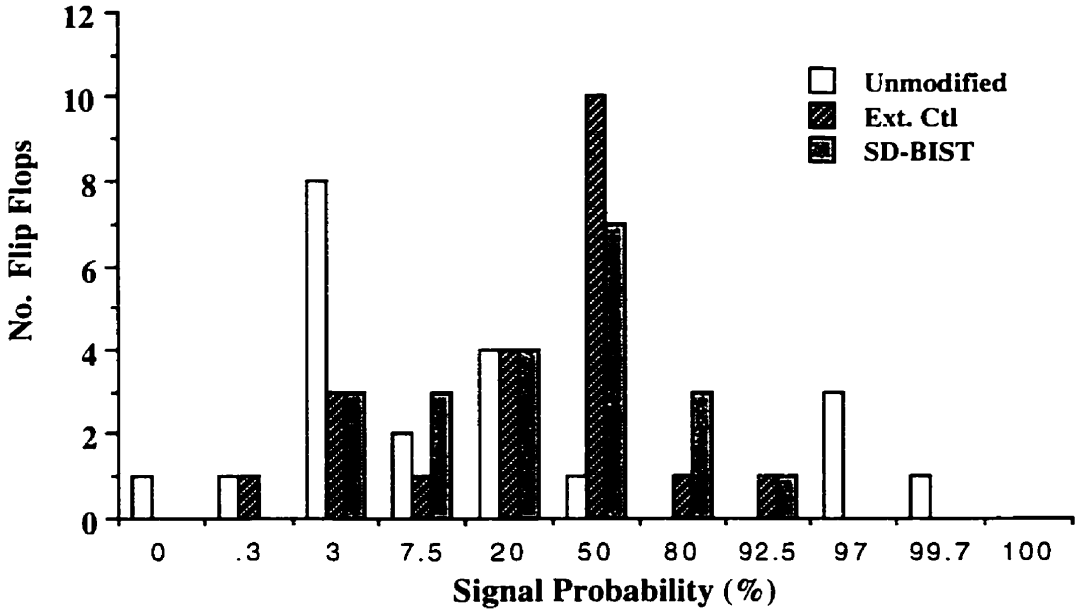


Figure B.6: s444 - Distribution of Flip Flop Biases

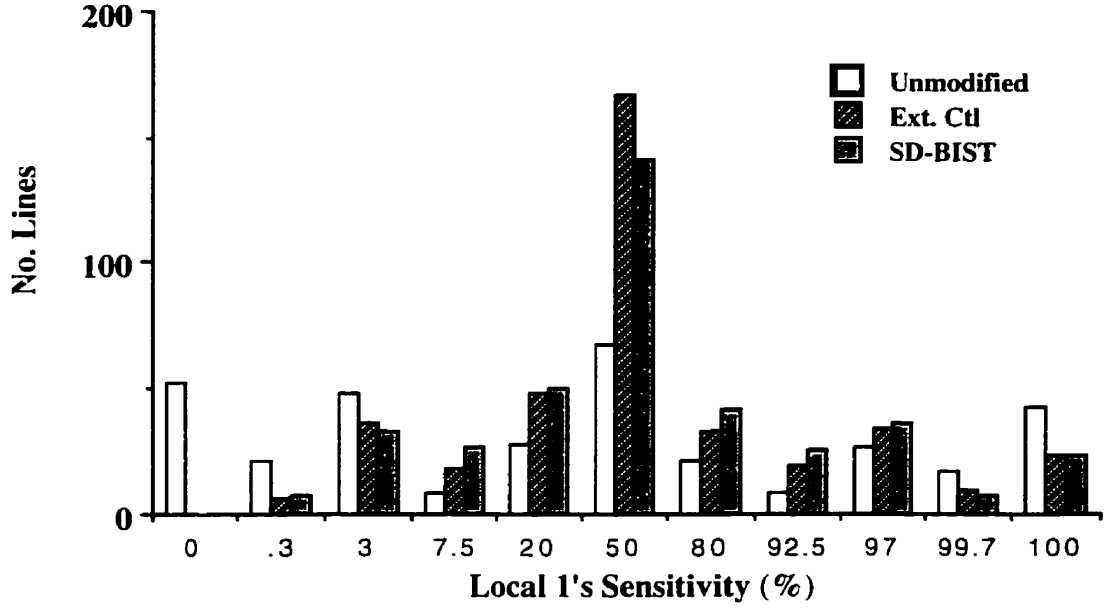


Figure B.7: s444 - OL_1 Distribution

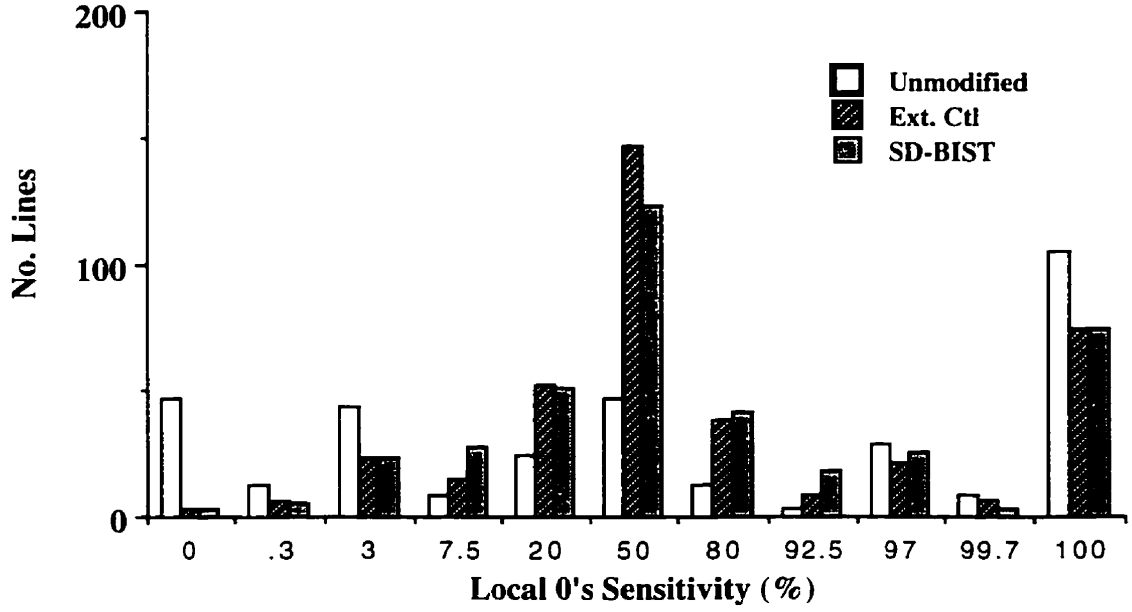


Figure B.8: s444 - OL_0 Distribution

Circuit - s526n

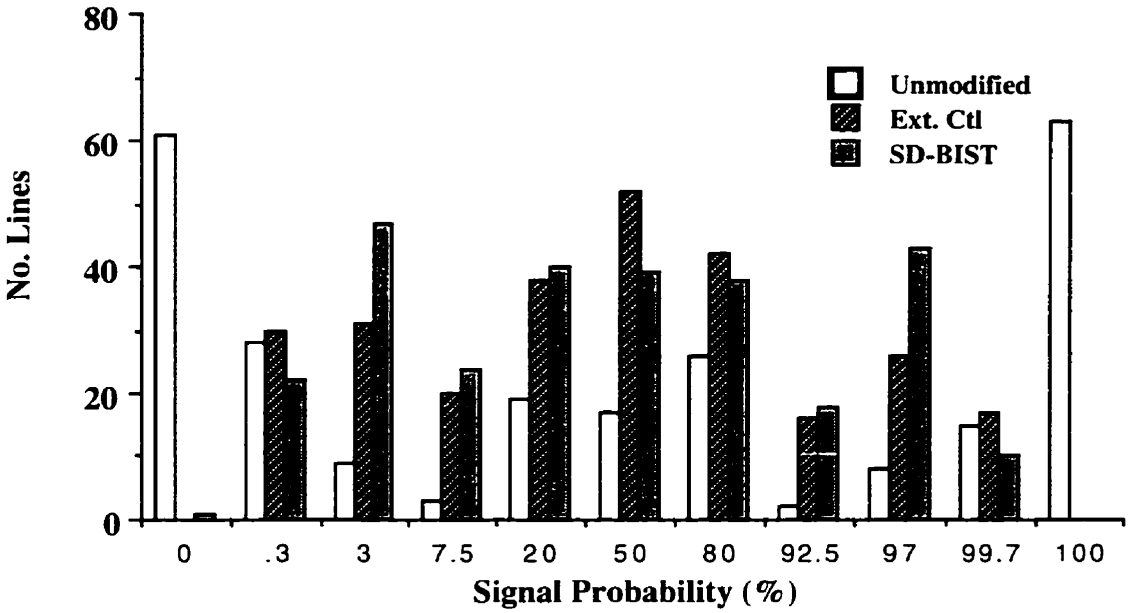


Figure B.9: s526n - Distribution of Line Biases

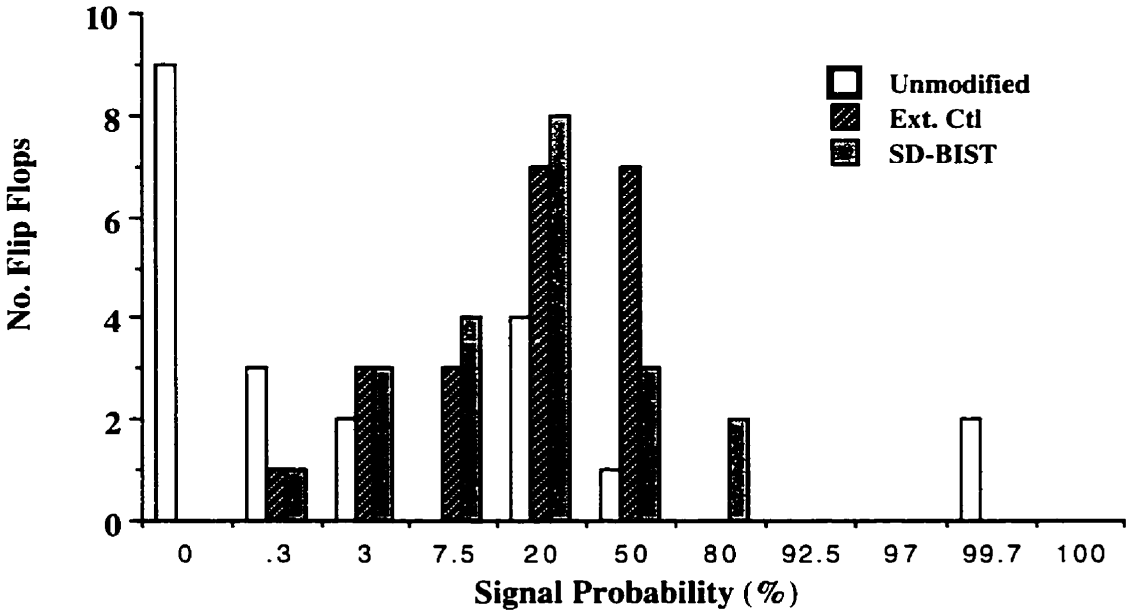


Figure B.10: s526n - Distribution of Flip Flop Biases

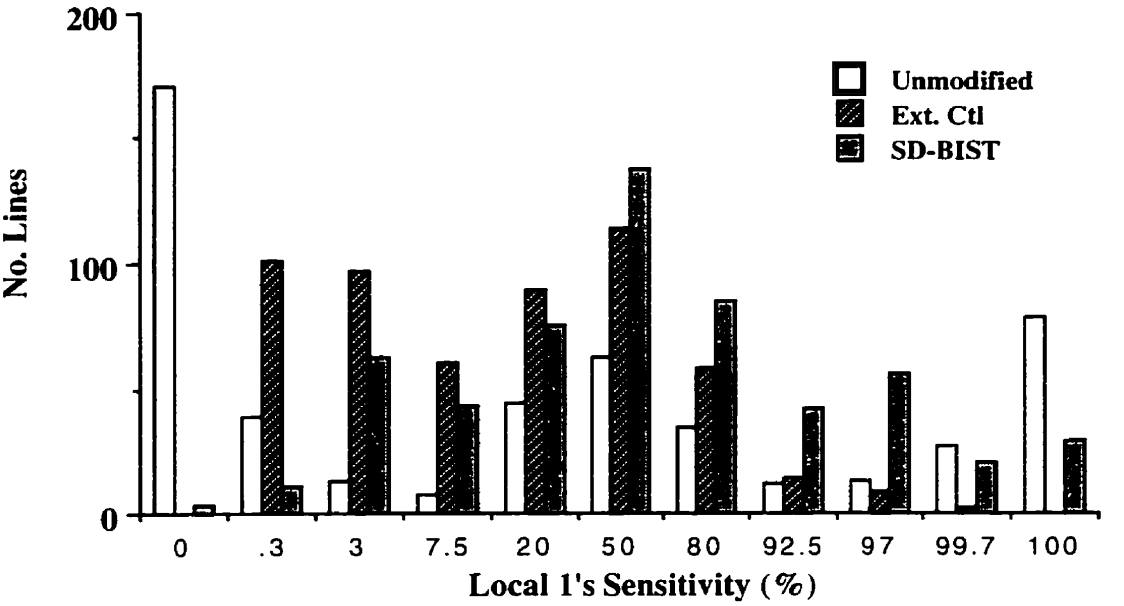


Figure B.11: s526n - OL_1 Distribution

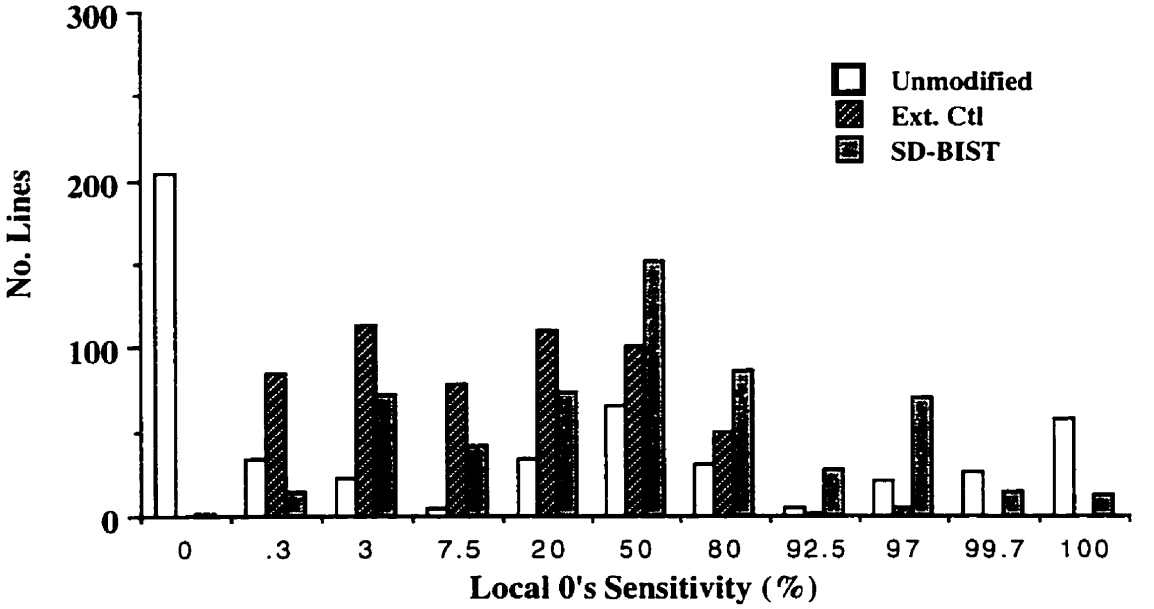


Figure B.12: s526n - OL_0 Distribution

Circuit - s641

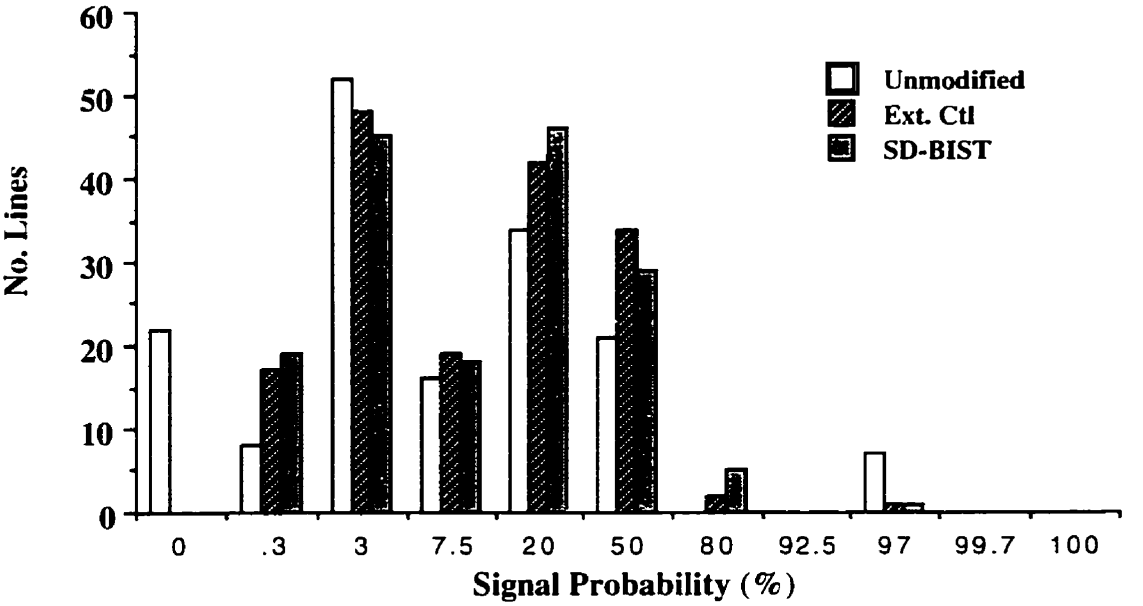


Figure B.13: s641 - Distribution of Line Biases

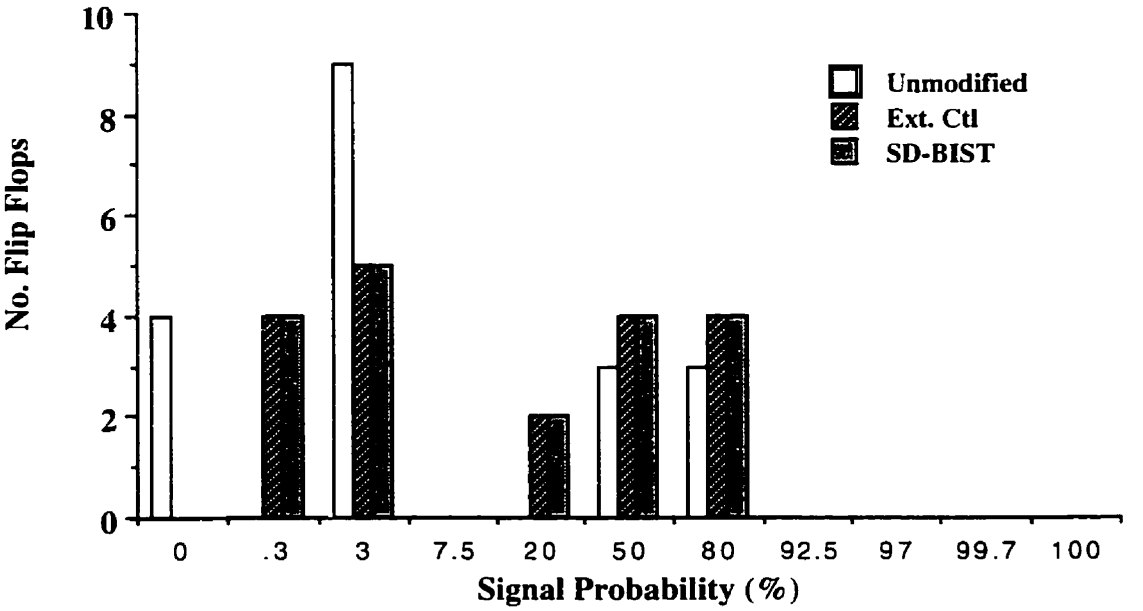


Figure B.14: s641 - Distribution of Flip Flop Biases

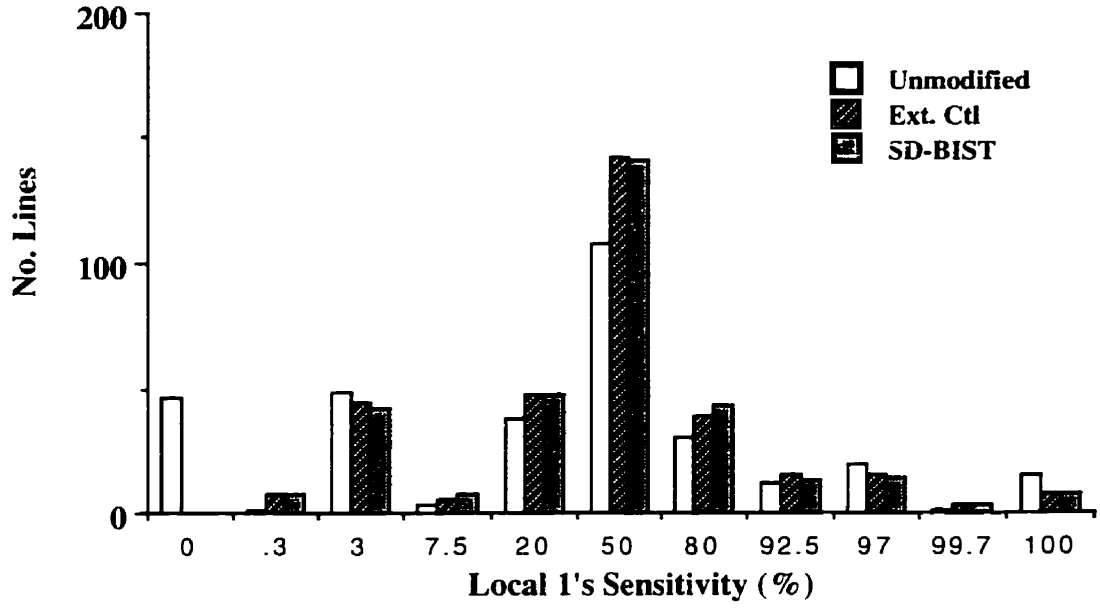


Figure B.15: s641- OL_1 Distribution

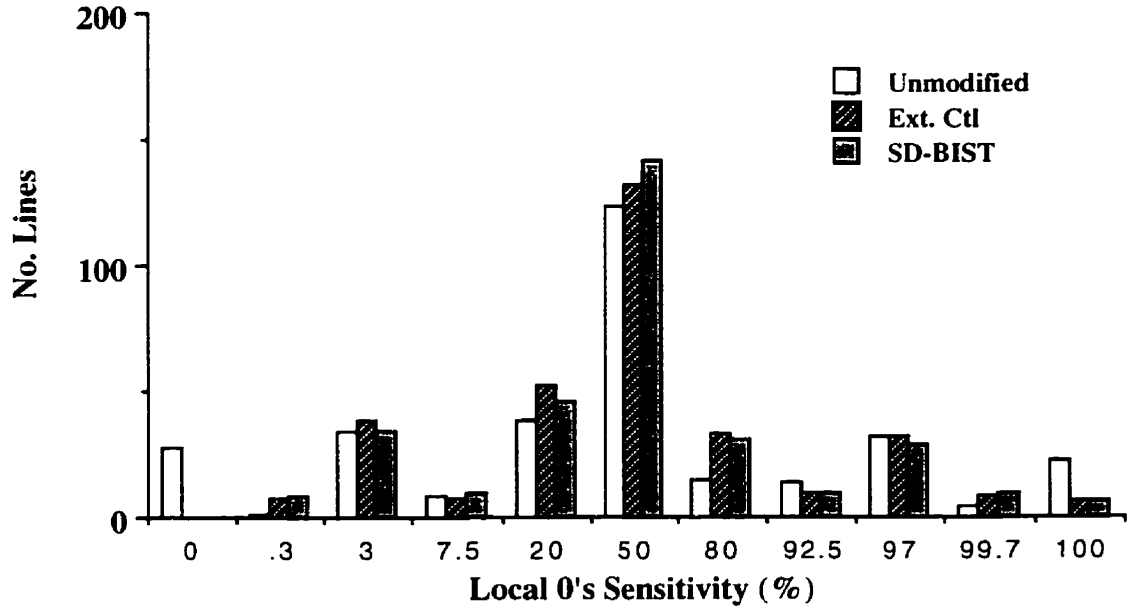


Figure B.16: s641 - OL_0 Distribution

Circuit - s820

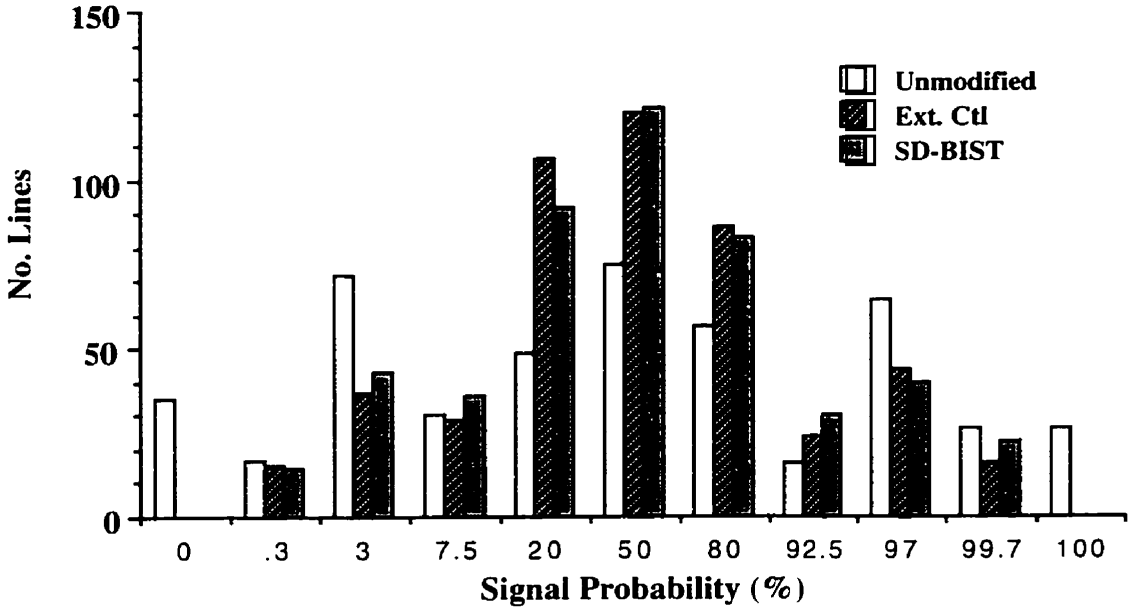


Figure B.17: s820 - Distribution of Line Biases

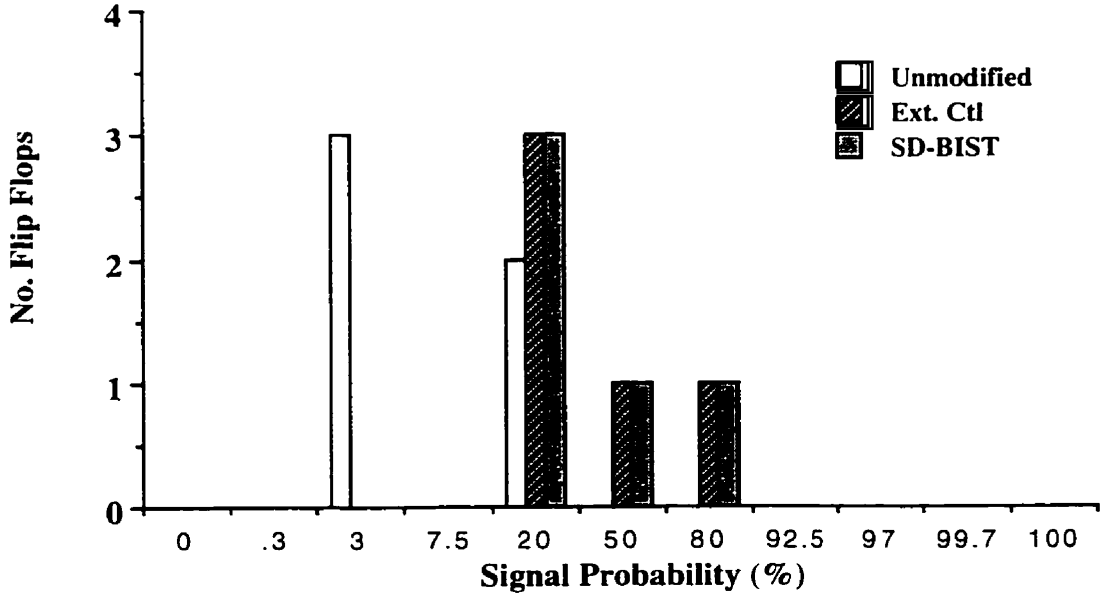


Figure B.18: s820 - Distribution of Flip Flop Biases

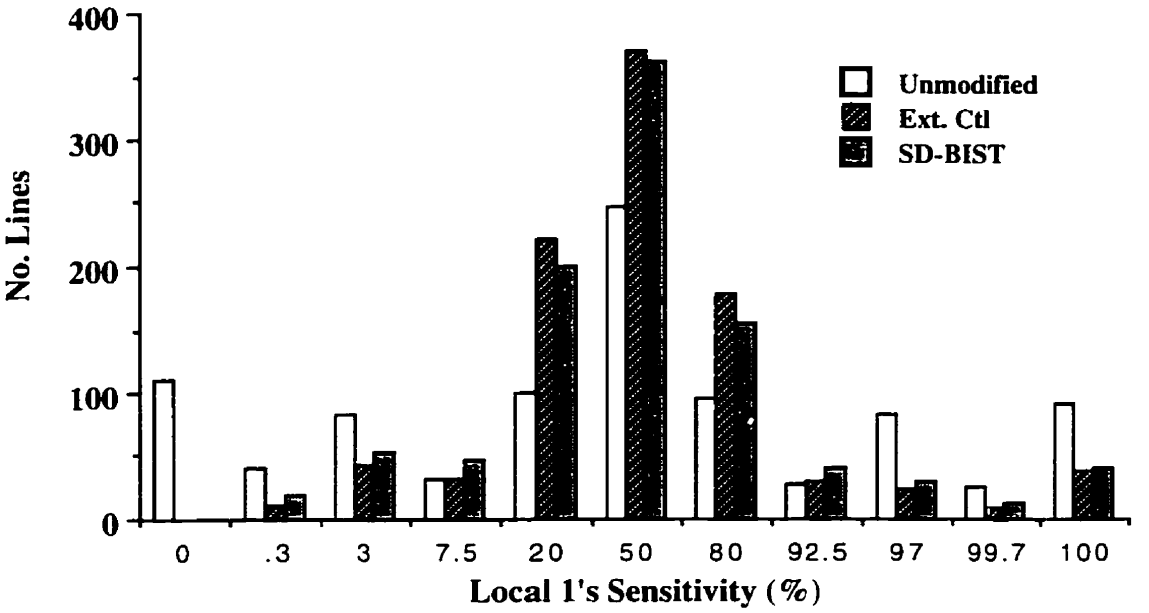


Figure B.19: s1423 - OL_1 Distribution

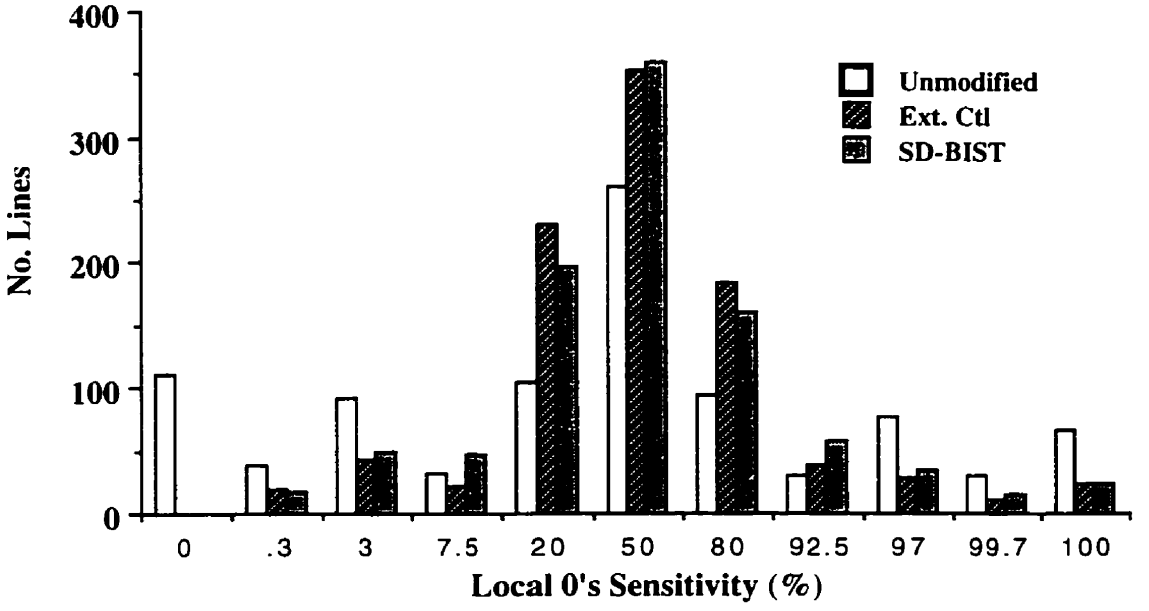


Figure B.20: s820 - OL_0 Distribution

Circuit - s832

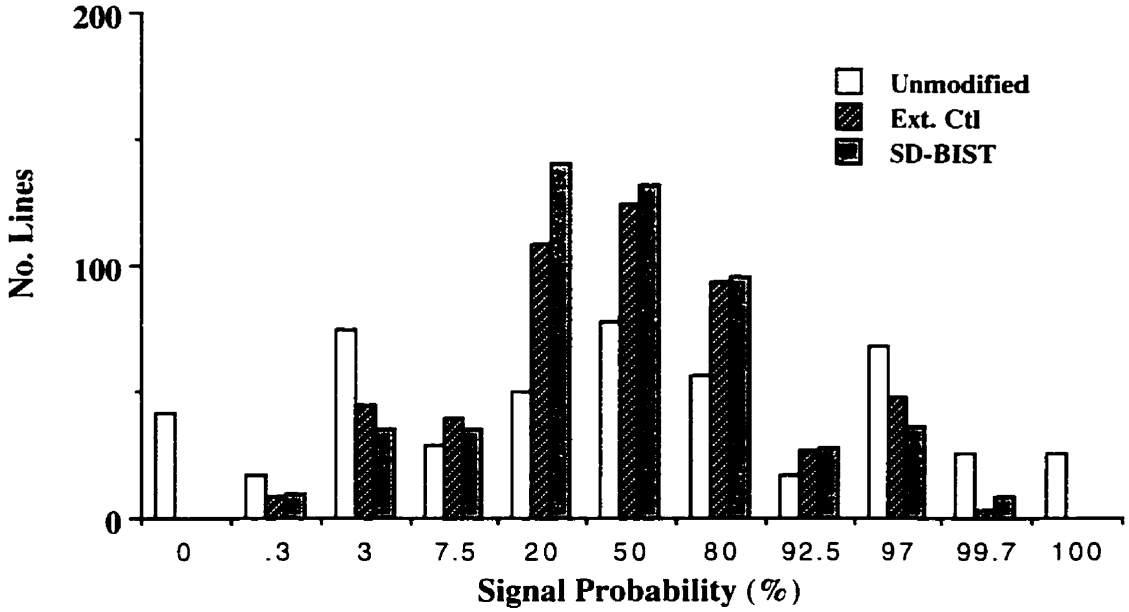


Figure B.21: s832 - Distribution of Line Biases

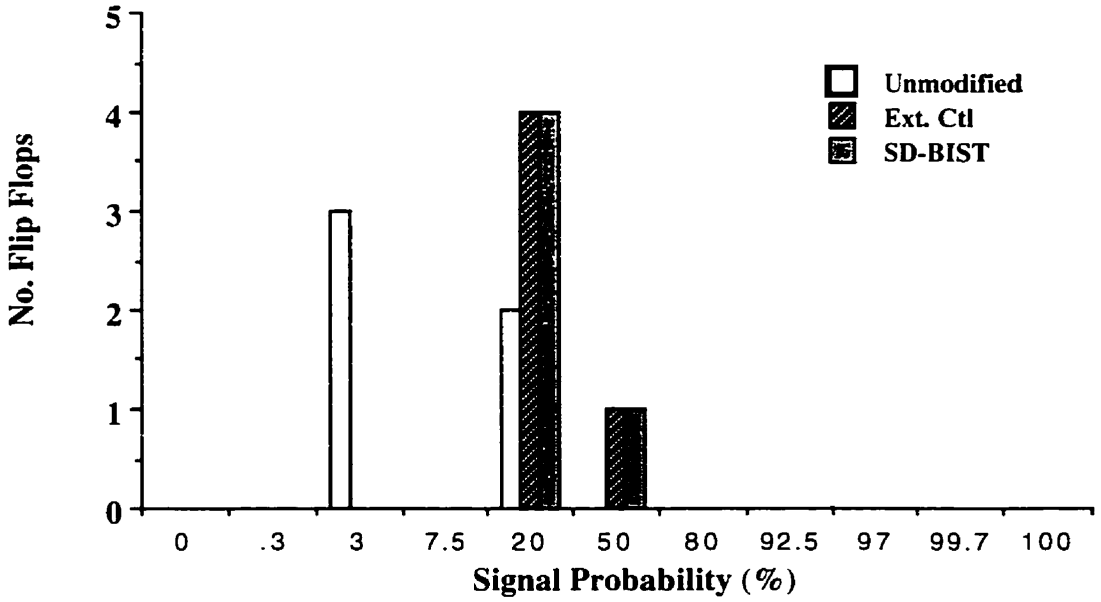


Figure B.22: s832 - Distribution of Flip Flop Biases

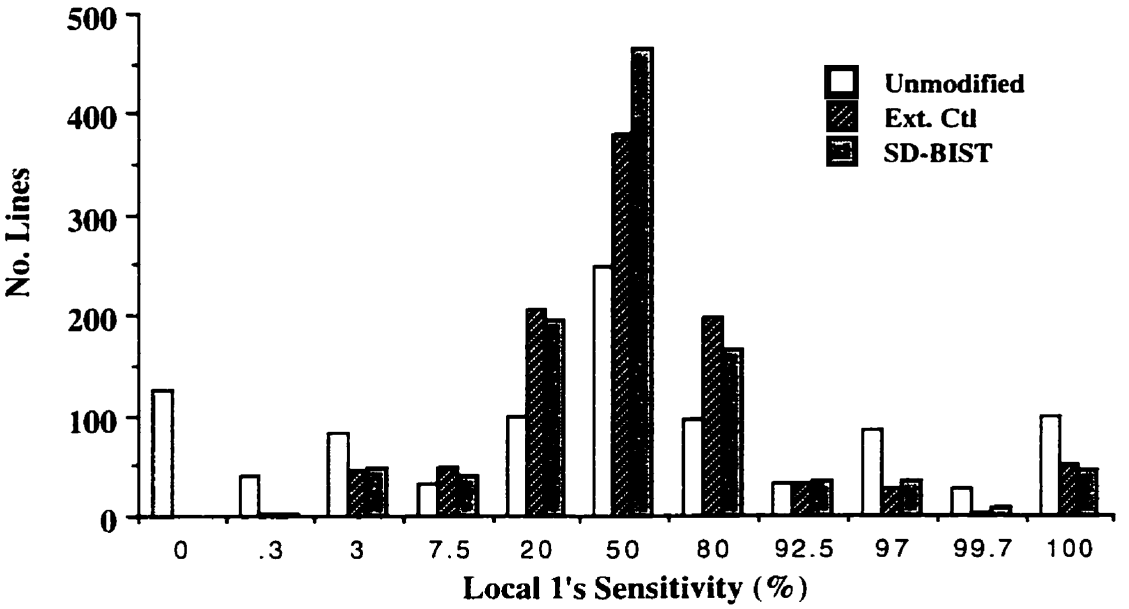


Figure B.23: s832 - OL_1 Distribution

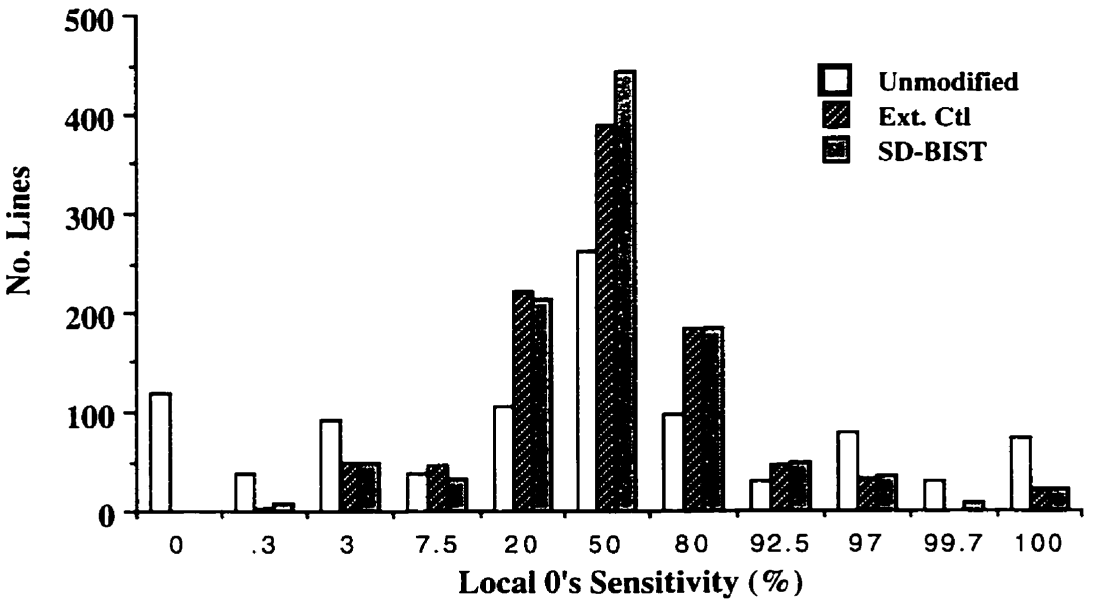


Figure B.24: s832 - OL_0 Distribution

Circuit - s838

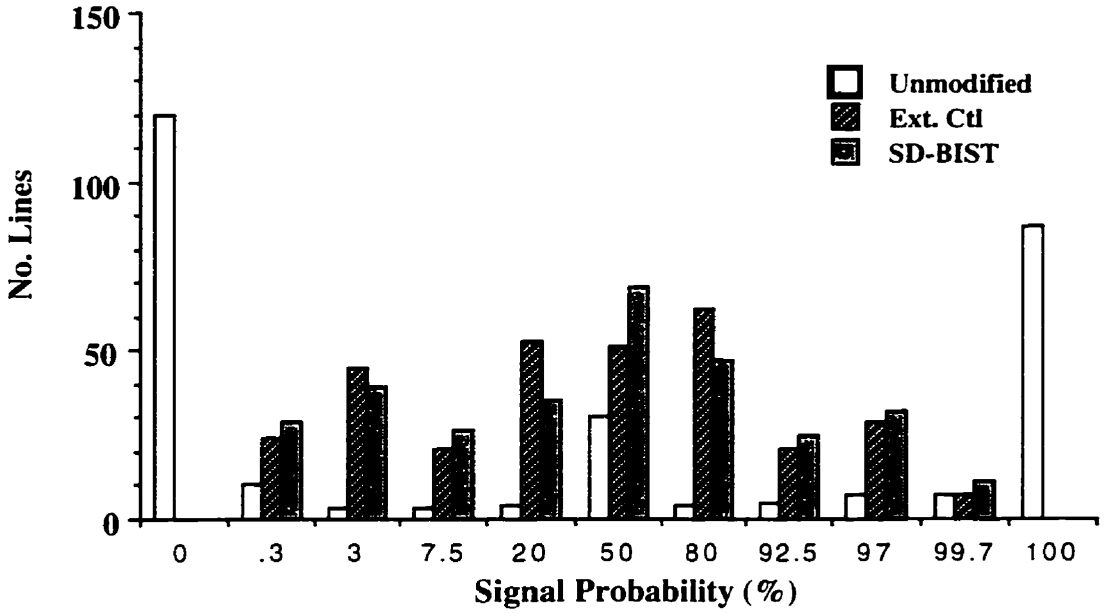


Figure B25: s838 - Distribution of Line Biases

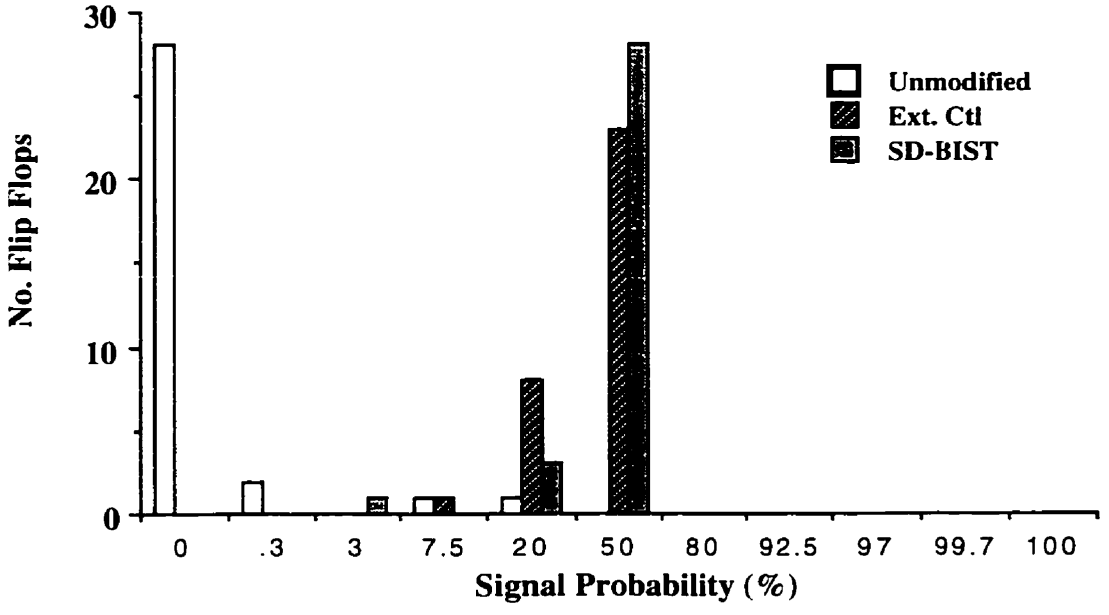


Figure B.26: s838 - Distribution of Flip Flop Biases

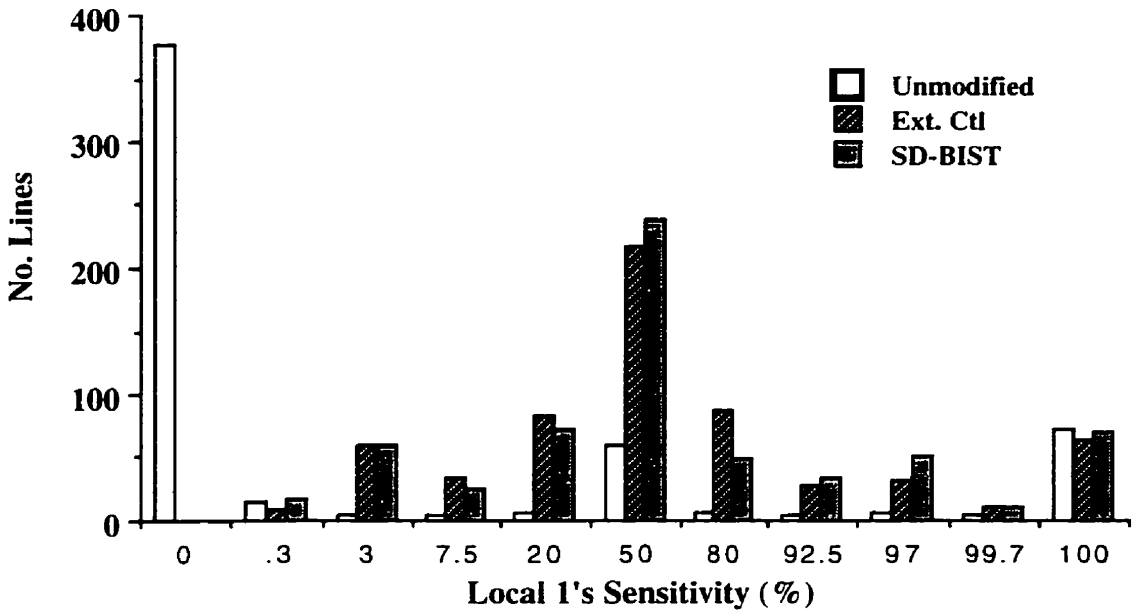


Figure B.27: s838 - OL_1 Distribution

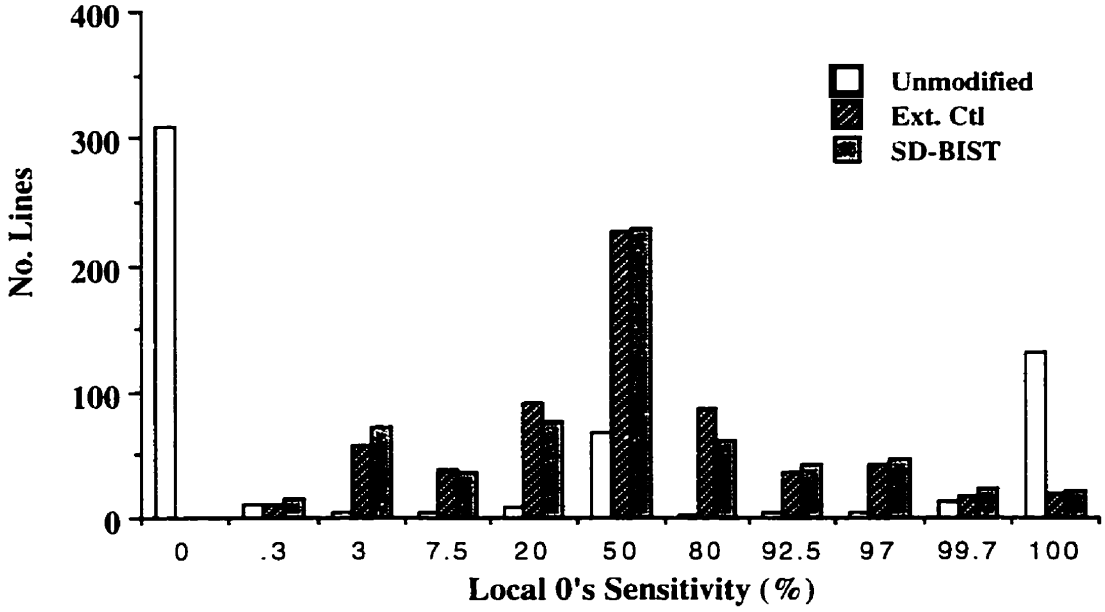


Figure B.28 : s838 - OL_0 Distribution

Circuit - s1423

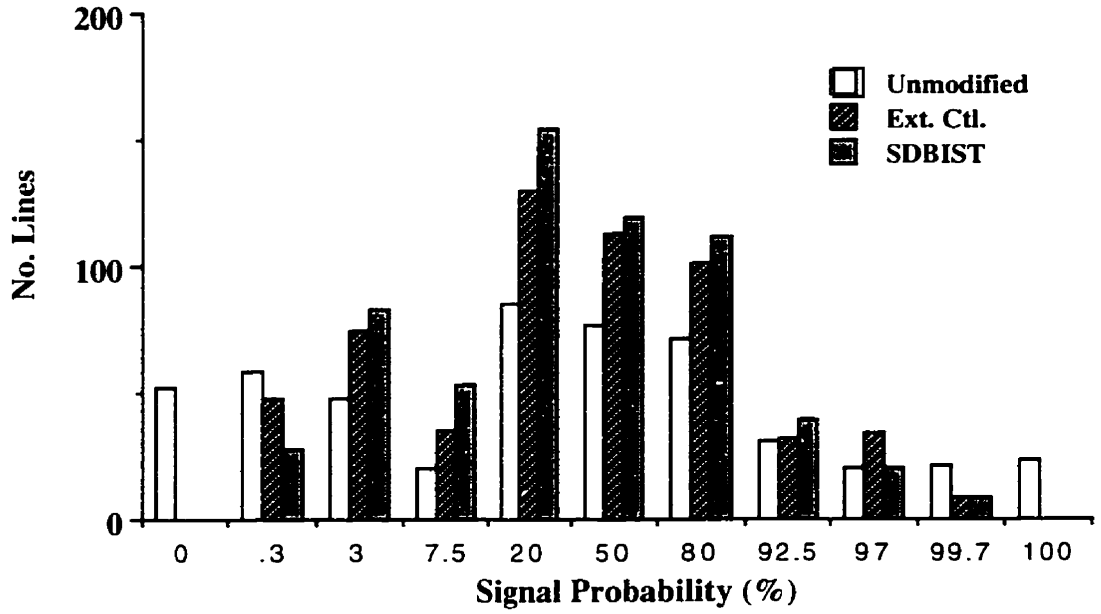


Figure B.29: s1423 - Distribution of Line Biases

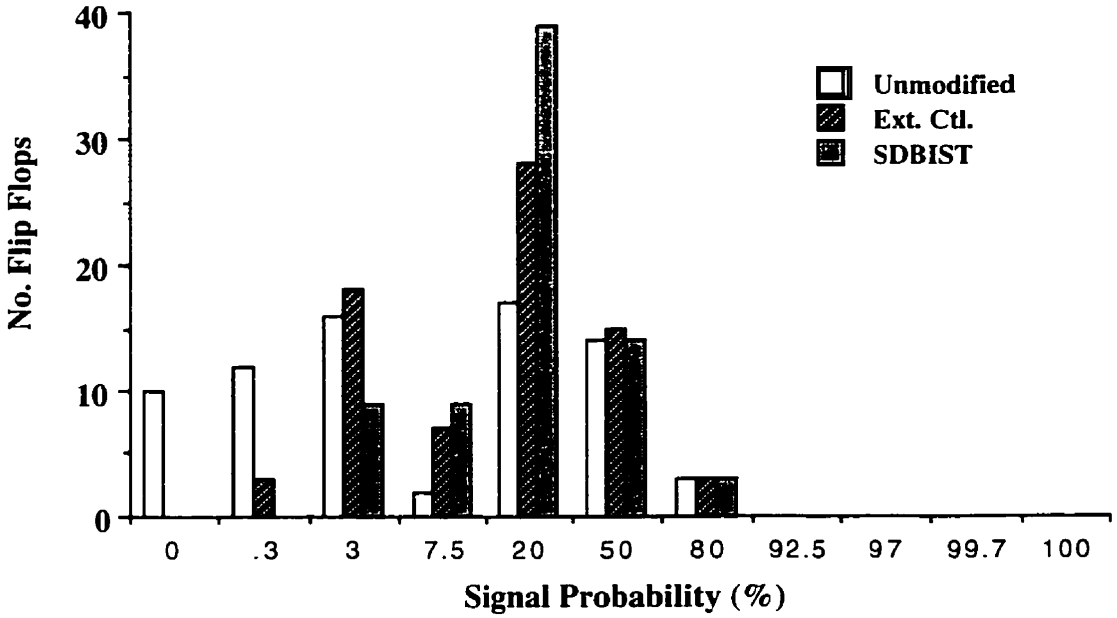


Figure B.30: s1423 - Distribution of Flip Flop Biases

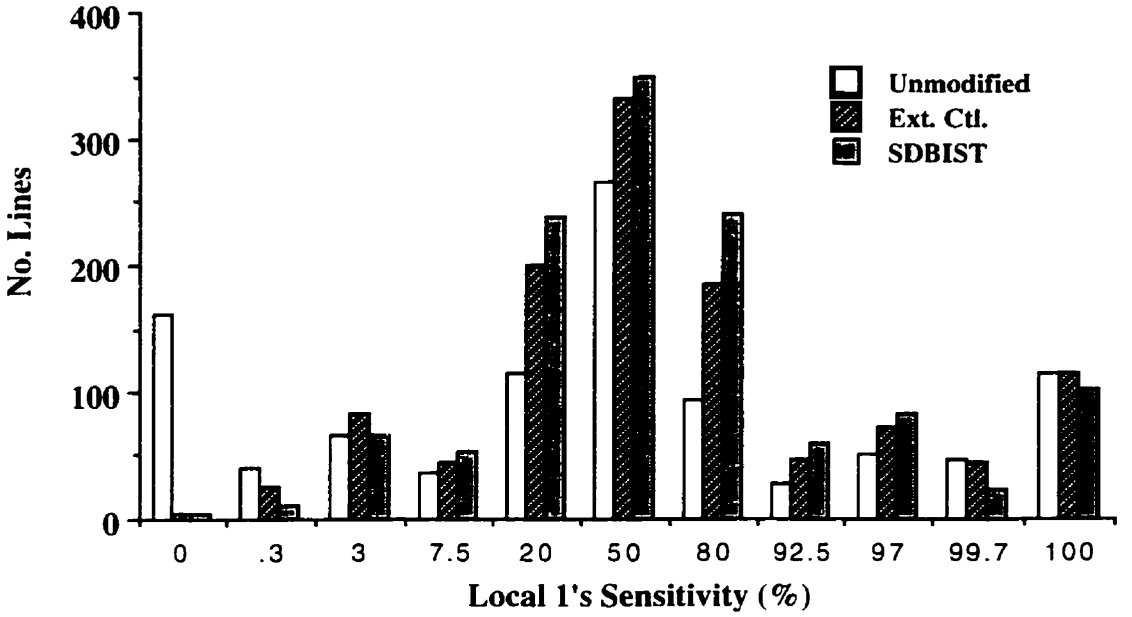


Figure B.31: s1423 - OL_1 Distribution

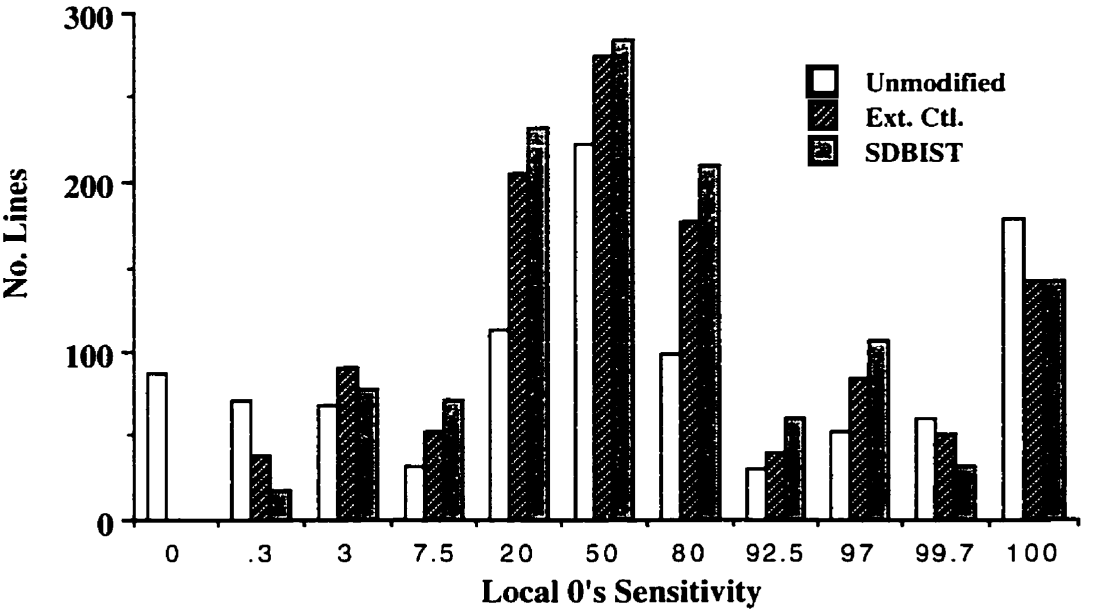


Figure B.32: s1423 - OL_0 Distribution

Circuit - 5378

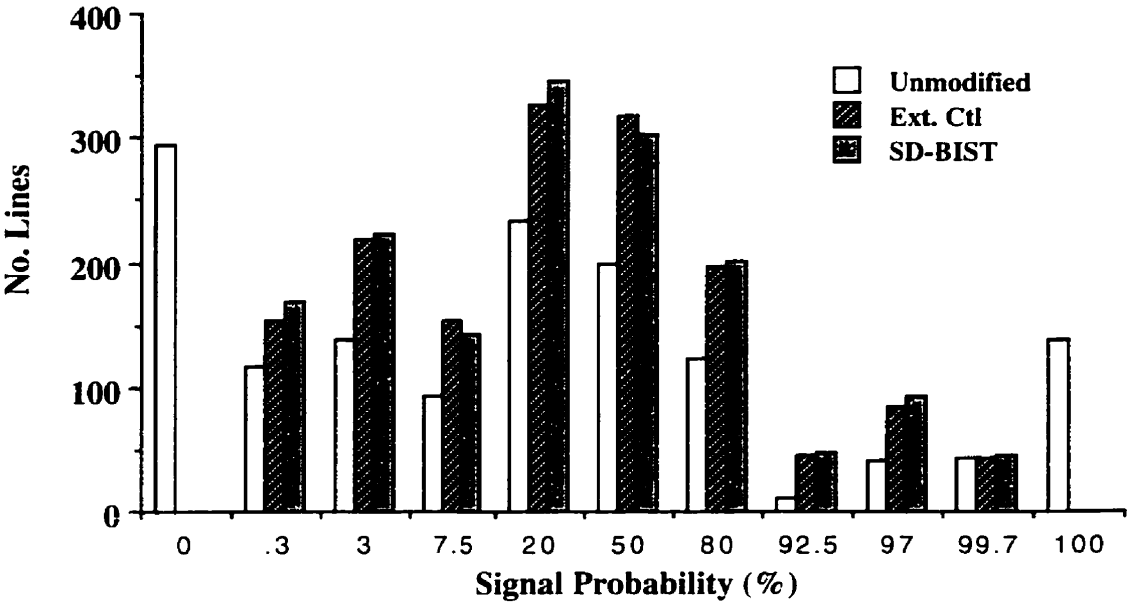


Figure B.33: s5378 - Distribution of Line Biases

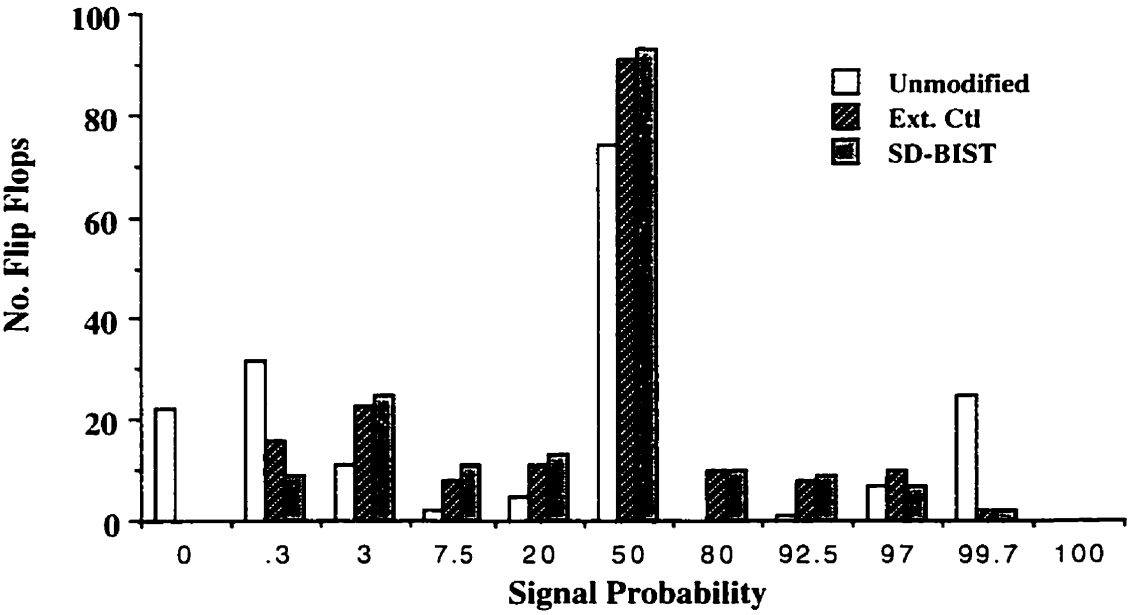


Figure B.34: s5378 - Distribution of Flip Flop Biases

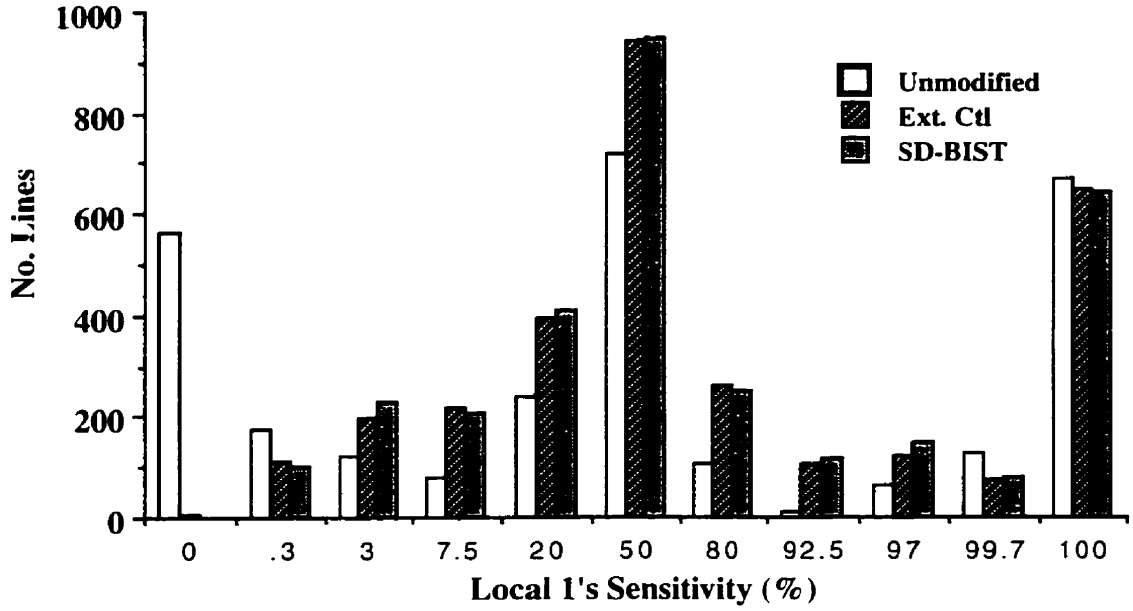


Figure B.35: s5378 - OL_1 Distribution

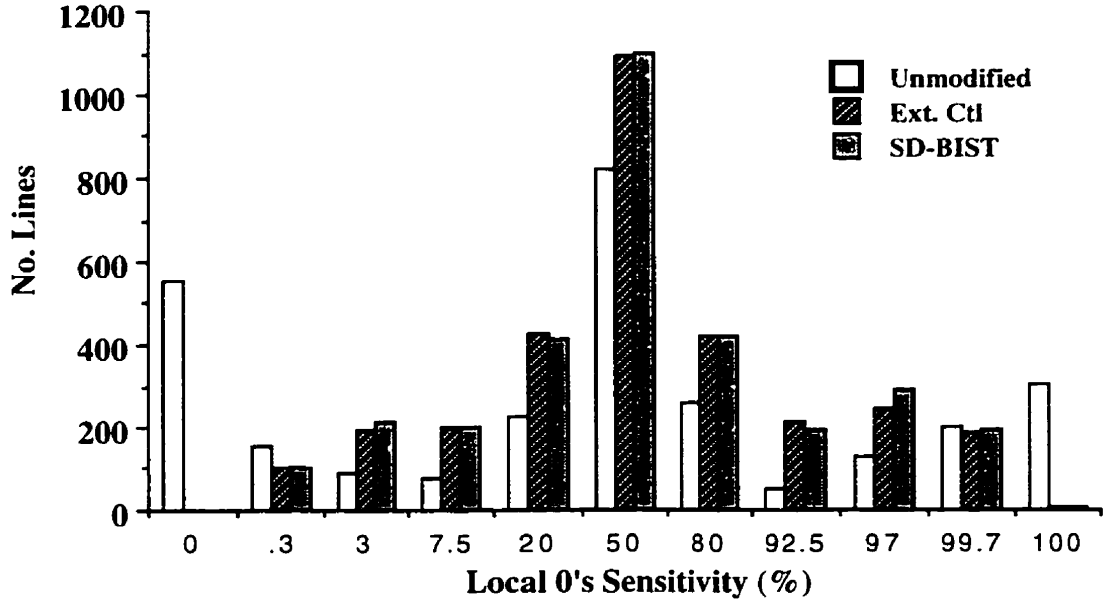


Figure B.36: s5378 - OL_0 Distribution

Circuit - s9234

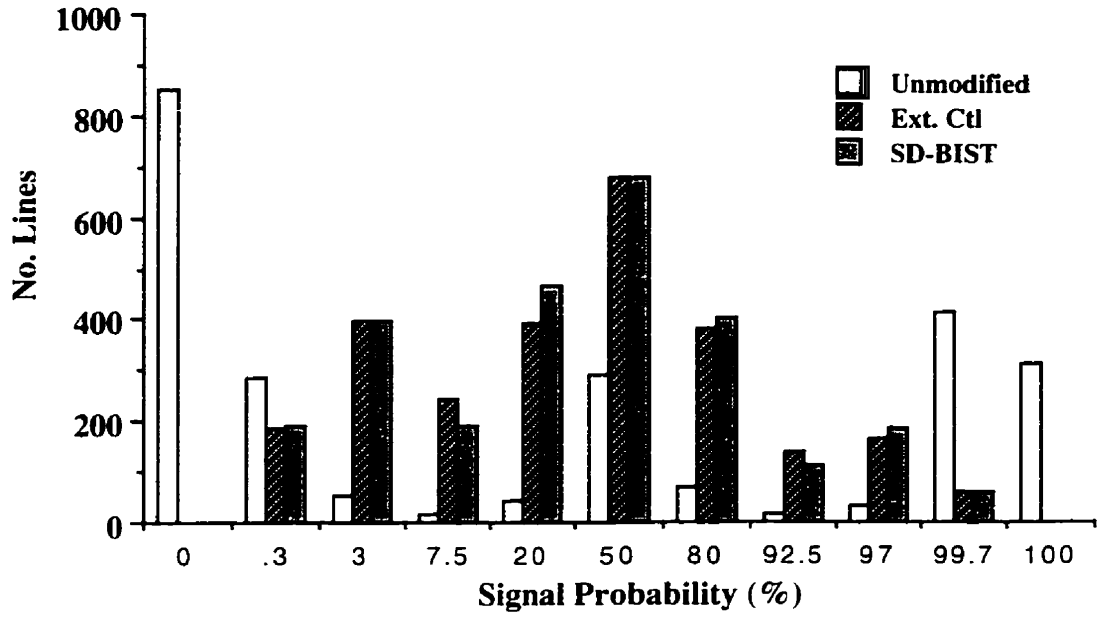


Figure B.37: s9234 - Distribution of Line Biases

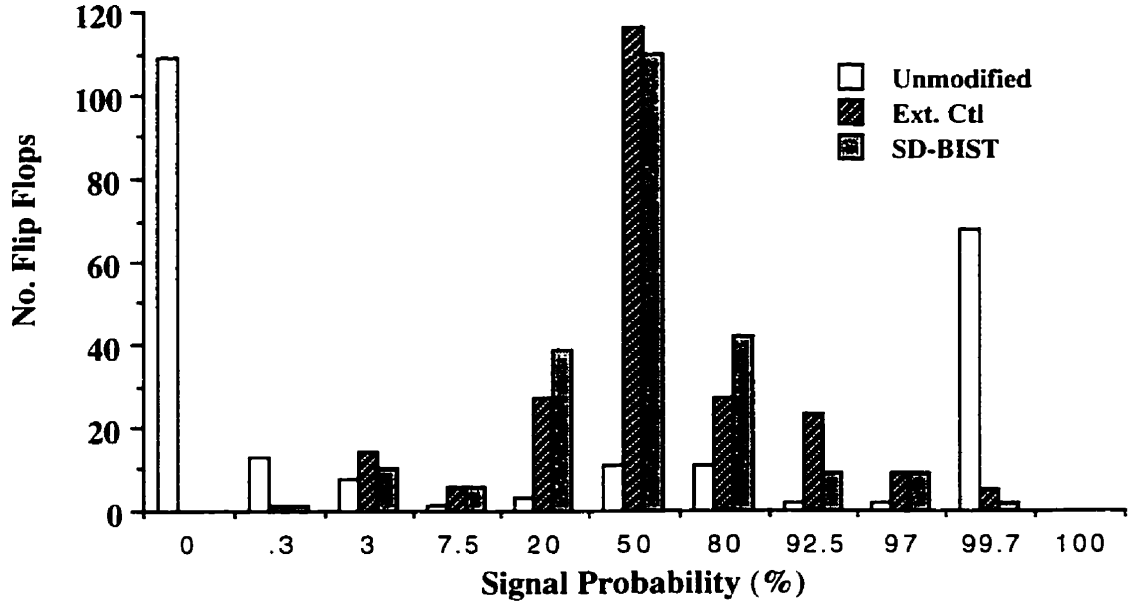


Figure B.38: s9234 - Distribution of Flip Flop Biases

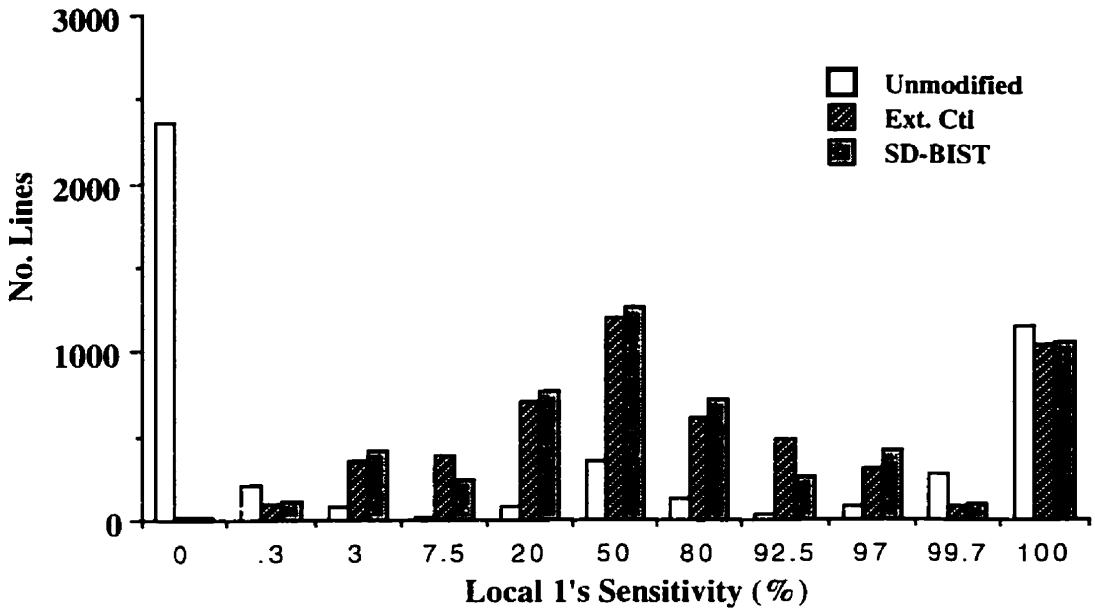


Figure B.39: s9234 - OL_1 Distribution

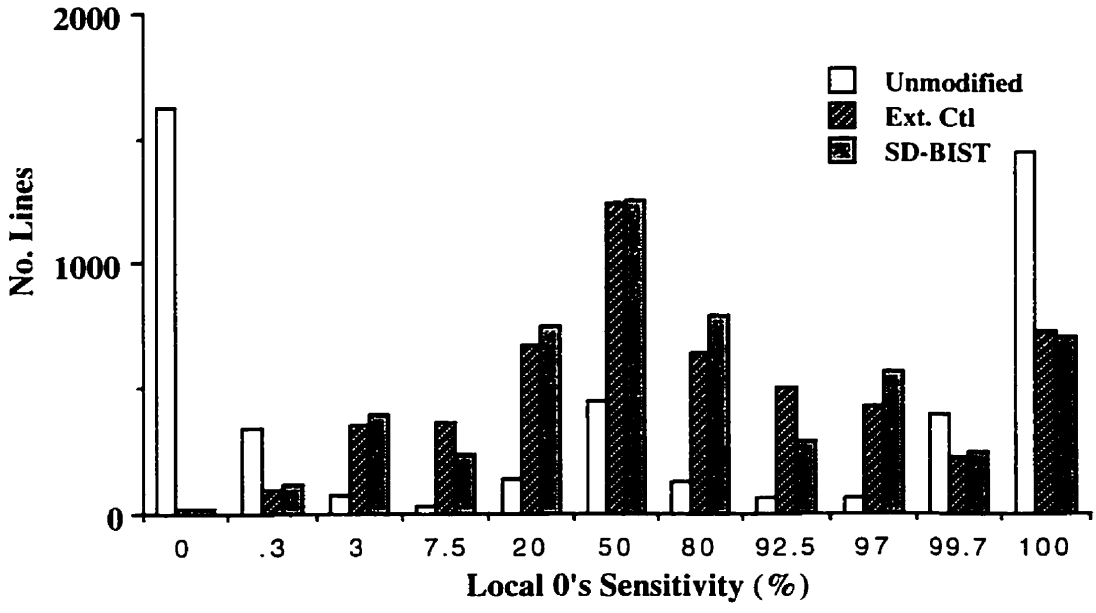


Figure B.40: s9234 - OL_0 Distribution

Circuit - s38417

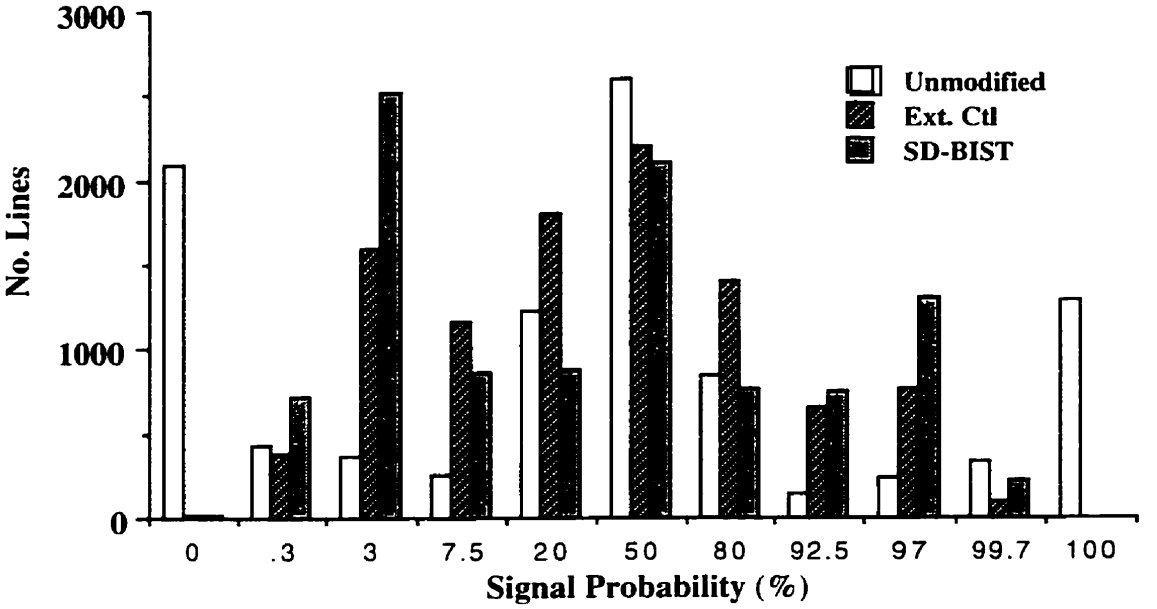


Figure B.41: s38417 - Distribution of Line Biases

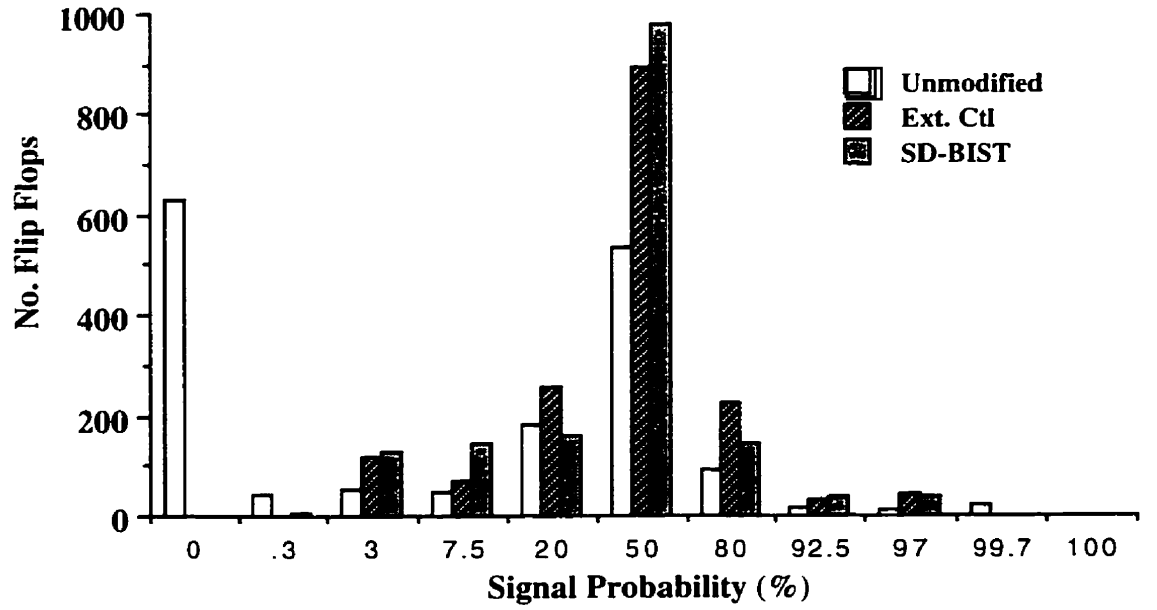


Figure B.42: s38417 - Distribution of Flip Flop Biases

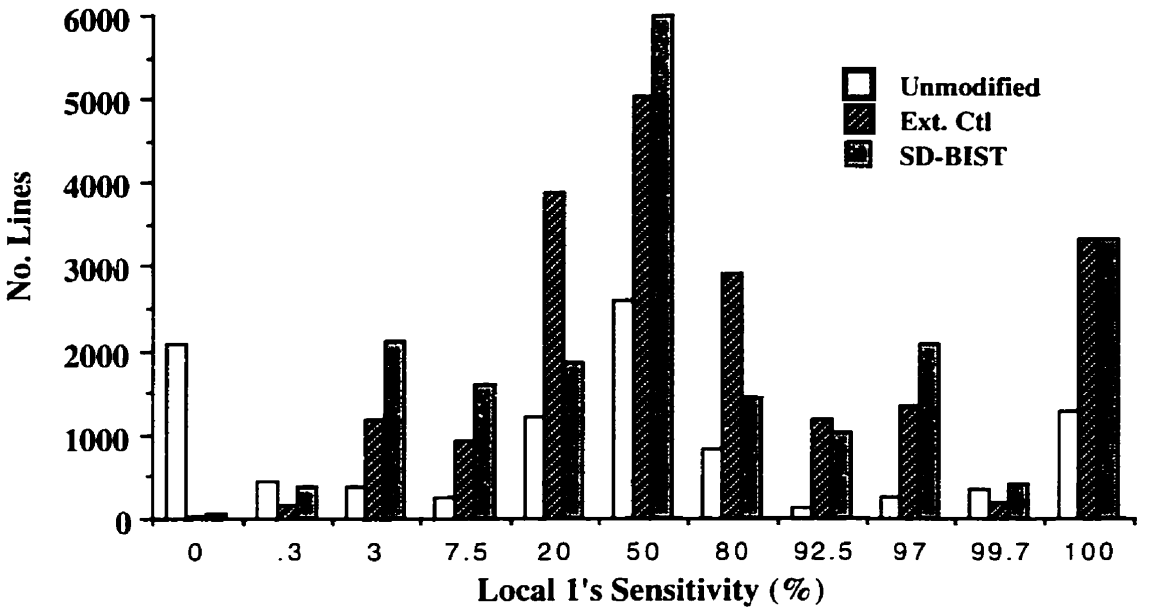


Figure B.43: s38417 - OL_1 Distribution

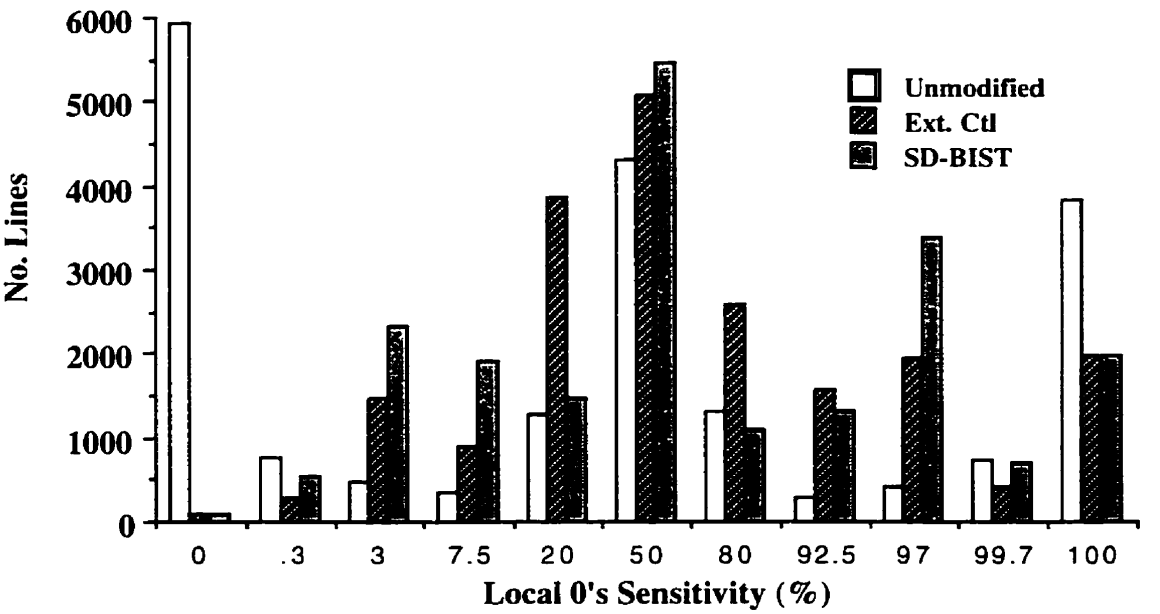


Figure B.44: s38417 - OL_0 Distribution

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.

[Abr84] M. Abramovici *et al.*, "Critical Path Tracing: An Alternative to Fault Simulation," *IEEE Design & Test of Computers*, Vol. 1, No. 1, pp. 83-93, Feb. 1984.

[Abr90] M. Abramovici, M.A. Bruer, A.D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, 1990.

[Abr92] M. Abromovici, K. Rajan, D. Miller, " FREEZE: A New Approach for Testing Sequential Circuits," *Proc. 29th Design Automation Conf.*, pp 22-25, 1992.

[Abr93] M. Abramovici, et al., "On selecting Flip Flops for Partial Reset," *Proc. IEEE Int'l. Test Conference*, pp. 1008-1012, 1993.

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

[Aga94] V.K. Agarwal, F. Muradali, B. Nadeau-Dostie, "A Scan Cell for Weighted Random Pattern Generation and Method of Operation" - *United States Patent 5,323,400* , June 1994...

[Agr81] V.D. Agrawal, "Sampling Techniques for Determining Fault Coverage in LSI Circuits," *Journal of Digital Systems*, Vol.5, No. 3, pp. 189-202, Fall 1981

[Agr82] V.D. Agrawal and M.R. Mercer, " Testability Measures - What Do They Tell Us?," *Proc. ITC. '82.*, pp. 391-396, 1982.

[Agr87] V.D. Agrawal, et al., "A Complete Solution to the Partial Scan Problem," *Proc. Int'l. Test Conference*, pp. 44-51, 1987.

[Agr89] V.D. Agrawal, K-T. Cheng, P. Agrawal, "A Directed Search Method for Test Generation Using a Concurrent Fault Simulator," *IEEE Trans. Comp. Aided Design*, pp. 131-138, 1989.

[Agr93] V.D. Agrawal, C. Kime, K.K. Saluja, "A Tutorial on Built-In Self-Test," *IEEE Design and Test*, pp. 73-82, March 1993.

[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*, pp 50-52, 1980.

[Arm72] D.B. Armstrong, "A Deductive Method for Simulating Faults in Logic Circuits," *IEEE Trans. on Computers*, Vol. C-21, pp. 464-471, May 1972.

[Aut92] E. Auth, J. Kastner, "Improved Test Pattern Generation for Sequential Circuits using Implicit Enumeration," *Proc. Int'l. Symp. Circuits and Systems '92*, pp. 1137-1140, 1992.

[Bar82] P.H. Bardell, W. H. McAnney, "Self-Testing of Multichip Logic Modules," *Proc. Int'l. Test Conference*, pp. 200-204, 1982.

[Bar84] P.H. Bardell, W. H. McAnney, "Parallel Pseudorandom Sequences for Built-In Self-Test," *Proc. Int'l. Test Conference*, pp.302-308, 1984.

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

[Bar89] P.H. Bardell, "Calculating the Effects of Linear Dependencies in m-Sequences Used as Test Stimuli," *Proc. Int'l. Test Conference*, pp. 252-256, 1989.

[Bar96] S. Barbagallo et al., "Scan Insertion for Low Design Impact," *Proc. 14th IEEE VLSI Test Symp.*, pp. 26-31, 1996

[Bas89] R.W. Basset et al., "Low Cost Testing of High Density Logic Components," *Proc ITC-89*, pp550-557, 1989.

[Ben75] N. Benowitz et al., "An Advanced Fault Isolation System for Digital Logic," *IEEE Trans. on Computers*, Vol. C-24, No. 5, pp. 489-497, May 1975.

[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.

[Brg84] F. Brglez, P. Pownall, R. Hum, "Applications of Testability Analysis: From ATPG to critical Delay Path Tracing," *Proc. ITC*, pp. 705-712, 1984.

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

[Brg89b] F. Brglez, D. Bryan, K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," *Proc. International Symp. on Circuits & Systems*, pp. 1929-1934, 1989.

[Bri86] A.J. Briers, K.A.E. Totton, "Random Pattern Testability By Fast Fault Simulation," *Proc. International Test Conference*, pp.274-281, 1986.

[Bry90] O. Brynestad, E. Aas, A.E. Vallestad, "State Transition Graph Analysis as a Key to BIST Fault Coverage," *Proc. Int'l. Test Conf.*, pp. 537-543, 1990.

[But92] K.M. Butler et al., "The Roles of Controllability and Observability in Design for Test," *Proc. IEEE VLSI Test Symp.*, pp. 211-216, 1992.

[Car82] W.C. Carter, "The Ubiquitous Parity Bit," *Proc. Fault Tolerant Computing Symp.* pp. 289-296, 1982.

[Cha94] S.C. Chakradhar, A. Balakrishnan, V.D. Agarwal, "An Exact Algorithm for Determining Partial Scan Flip Flops," *Proc. 31st Design Automation Conference*, 1994.

[Cha95] K. Chakrabarty, B.T. Murray, J.P. Hayes, "Optimal Space Compaction of Test Responses," *Proc. Int'l Test Conference*, pp. 834-843, 1995

[Che88] W-T. Cheng, " The BACK Algorithm for Sequential Generation," *Proceedings International Conference on Computer Design*, pp. 66-69, 1988.

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

[Che89a] W-T Cheng, M-L. Yu, "Differential Fault Simulation - A Fast Method Using Minimal Theory," *Proc. 26th Design Automation Conference*, pp. 424-428.

[Che89c] C.H. Chen, P.R. Menon, "An Approach to Functional Level Testability Analysis," *Proc. Int'l. Test Conf.*, pp. 373-379, 1989.

[Che89d] K-T Cheng, V.D. Agrawal, "Design of Sequential Machines for Efficient Test Generation," *Proc. Int'l. Conf. Computer-Aided Design*, pp. 358-361, 1989.

[Che90] K-T Cheng, J-Y Jou, "Functional Test Generation for Finite State Machines," Proc. ITC '90, pp.162-166, 1990.

[Che90b] K-T Cheng, V.D. Agrawal, "A PArtial Scan Method for Sequential Circuits," *IEEE Trans. Computers*, Vol. 39, No. 4, pp. 544-548, April 1990.

[Che91] K-T. Cheng, "On Removing Redundancy in Sequential Circuits," *Proc. 28th Design Autoamtion Conf.*, pp. 164-169, 1991.

[Che92] K-T Cheng, "Recent Advances in Sequential Test Generation," Proc. IEEE VLSI Test Symp. '92, pp. 241-246, 1992.

[Che92b] K-T Cheng, H-K Ma, "On The Overspecification Problem in Sequential ATPG Algorithms," *Proc. 29th Design Automation Conference*, pp. 16-21, 1992.

[Che95] K-T. Cheng, C-J. Lin, "Timing Driven Test Point Insertion for Full-Scan and Partial Scan BIST," *Proc. Int'l. Test Conference*, 506-514, 1995.

[Chi90] V. Chickermane, J.H. Patel, "An Optimization Based Approach to the Partial Scan Design Problem," *Proc. Int'l. Test Conference*, pp. 377-386, 1990.

[Chi93] V. Chikermane, E.M. Rudnick, P. Banerjee, J.H. Patel, " Non-Scan Design for Testability Techniques for Sequential Circuits," *Proc. 30th Design Automation Conference*, pp. 236-24, June 1993.

[Cor94] F. Corno, P. Prinetto, M. Sonza-Reorda, "Making the Circular Self-Test Path Effective for Real Circuits," *Proc. Int'l. Test Conf.*, pp. 949-957, 1994.

[Chu89] C. Chuang, A. Gupta, "The Analysis of Parallel BIST by the Combined Markov Chain," *Proc. Int'l. Test Conference*, pp. 337-343, 1989.

[Dav76] R.David, G. Blanchet, "About Random Fault Detection of Combinational Networks," *IEEE Trans. Comp.*, pp. 659-664, 1976.

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

[Dea94] C.A. Dean, Y. Zorian, "Do You Practice Safe Test? What We Found Out About Your Habits," *Proc. International Test Conference*, pp 887-892, 1994.

[DeM94] G. DeMicheli, *Synthesis and Optimization of Digital Circuits*, McGraw Hill Electrical and Computer Engineering Series, 1994.

[Dev89] S. Devadas et al., "A Synthesis & Optimization procedure for Fully & Easily Testable Sequential Machines," *IEEE Trans CAD.*, Vol. 8, No. 10, pp. 110, Oct 1989.

[Dev90] S. Devadas et al., "Irredundant Sequential Machines Via Optimal Logic Synthesis ," *IEEE Trans on CAD*, Vol 9. No. 1, Jan 1990.

[Dev91] S. Devadas, K. Keutzer, "A Unified Approach to the Synthesis of Fully Testable Sequential Machines," *IEEE Trans on CAD*, Vol. 10, No. 1, pg. 39, Jan 1991

[Duf91] C.Dufaza, G. Gambon, "LFSR-Based Deterministic and Pseudo-Random Test Pattern Generator Structures," *Proc. European test Conference*, pp.27-34, 1991.

[Eic78] E.B. Eichelberger, T.W. Williams, "A Logic Design Structure for LSI Testability," *J. Design Automat. Fault Tolerant Comp.*, Vol. 2, pp. 165-178, May 1978.

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

[Eld59] R.D. Eldred, "Test Routines Based on Symbolic Logical Statements," *Journal of the ACM*, Vol. 6 pp. 33-36, 1959.

[Esc90] B. Eschermann, H-J Wunderlich, "Optimized Synthesis of Self-Testable Finite State Machines," *20th Fault Tolerant Computing Symp.*, pp. 390-397, 1990.

[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

[Fox77] M.A. Fox, "Test Point Condensation in the Diagnosis of Digital Circuits," *Proc. IEE.* Vol. 124, No.2, pp. 89-94, Feb.,1977March 1986.

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

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

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

[Gho90] A. Ghosh, S. Devadas, A.R. Newton, "Sequential Test Generation at the Register-Transfer and Logic Levels", *Proc 27th DAC* , pp. 580-586, June 1990.

[Gho91] A. Ghosh, S. Devadas, A.R. Newton, "Test Generation and Verification for Highly Sequential Circuits," *IEEE Transactions on Computer-Aided Design*, pp. 652-657, May 1991.

[Gho92] A. Ghosh et al., "Estimation of Average Switching Activity in Comobinational and Sequential Circuits," *Proc. 29th Dessign Automation Conf.* , pp.253-259, 1992

[Gho92b] A. Ghosh, S. Devadas, A. R. Newton, *Sequential Logic Testing and Verification*, Kluwer Academic Publishers, 1992.

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

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

[Gou91] N. Gouders, R. Kaibel, "PARIS: A Parallel Pattern Fault Simulator for Synchronous Sequential Circuits," *Proc. Int'l. Conf. On Computer-Aided Design*, pp. 542-545, 1991.

[Gra79] J. Grason, "TIMEAS, A Testability Measurement Program," *Proc. 16th Design Automation Conference*, pp. 156-161, 1979.

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

[Gre94] D.A. Greene, "When Does It Make ¢ to Give Up Physical Test Access?" *Proc. International Test Conference*, pp 111-119, 1994.

[Gun90] H. Gundlach, K-D. Muller-Glaser, "On Automatic Test Point Insertion in Sequential Circuits," *Proc. Int'l. Test Conference*, pp. 1072-7079, 1990.

[Gup90] R.Gupta, R. Gupta, M.A. Breuer, "The Ballast Mathodology for Structured Partial Scan Design," *IEEE Trans. Computers*, Vol. 39, No. 4, pp. 538-544, April 1990.

[Ham91a] N.B. Hamida, B. Kaminska, "Hierarchical Functional Level Testability Analysis," *Proc. ETC. '91*, pp. 327-332, 1991.

[Ham91b] N.B. Hamida, B. Kaminska, "A Uniform Testability Measure Representation for Sequential Circuits," *Proc. Int'l. Symp. on Circuits and Systems*, pp. 327-332, 1991.

[Har89] M. Harihara, P.R. Memon, "Identification of Undetectable Faults in Combinational Circuits," *Proc. Int'l. Conf. Computer Design*, pp. 290-293, 1989.

[Hay74] J.P. Hayes, "Test Point Placement to Simplify Fault Detection," *Fault Tolerant Computing Symp.*, pp. 73-78, 1974.

[Hay76] J.P. Hayes, "Transition Count Testing of Combinational Logic Circuits," *IEEE trans. on Computers*, pp.Vol. C-25, pp.613-620, June 1976.

[Hat92] K. Hatayama, K. Hikone, M. Ikeda, T. Hayashi, "Sequential Test Generation Based on Real-Valued Logic Simulation," *Proc. Int'l. Test Conf.*, pp. 41-48, 1992.

[Hel95] S.Hellebrand et. al., "Built-In Self-Test for Circuits with Scan Based on Reseeding of Multiple Polynomial Linear Feedback Shift Registers," *IEEE Transactions on Computers*, pp.Vol. C-44, Feb.1995.

[Hen64] F.C. Hennie, "Fault Detecting Experiments For Sequential Circuits," *Proc. 5th Annual Symposium on Switching Circuit Theory and Logical Design*, pp. 95-110, Nov. 1964.

[Hil77] Fredrick J. Hill, Ben Heuy, "SCIRTSS: A Search System for Sequential Circuit Test Sequences," *IEEE Trans. Comp.*, pp. 490-502, May 1977.

[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.

[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.

[Hud89] R.V. Hudi, S.C. Seth, "Testability Analysis of Synchronous Sequential Circuits Based on Structural Data," *Proc. Int'l. test Conf.*, pp. 364-372, 1989.

[Hui88] L.M. Huisman, "The Reliability of Approximate Testability Measures," *IEEE Design and Test, Vol. 5*, pp. 57-67, Dec. 1988.

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

[Iba75] O.H. Ibarra, S.K. Sahni, "Polynomially Complete Fault Detection Problems," *IEEE Trans.Comp.* Vol. C-24, pp. 242-249, March 1975.

[Iye89] V.S Iyengar & D. Brand, "Synthesis of Pseudo-Random Pattern Testable Designs," *Proc. International Test Conference*, pp. 501-508, 1989.

[Iye94] M.A. Iyer, M. Abramovici, "Sequentially Untestable Faults Identified Without Search (Simple Implications Beat Exhaustive Search!)," *Proc. International Test Conference*, pp. 259-266, 1994.

[Jai84] S. Jain, V. Agarwal, "STAFAN: An Alternative Approach to Fault Simulation," *Proc. DAC '84*, pp. 18-23, 1984.

[Kap94] B. Kapoooor, "Improving the Accuracy f Circuit Activity Measurement," *Proc. 31st Design Automation Conf.*, pp. 734-739, 1994.

[Kas96] M. Kasaab, Ph.D Thesis, McGill University 1996.

[Kel89] T.P. Kelsey, K.K. Saluja, "Fast Test Generation for Sequential Circuits," *Proc. International Conference on Computer-Aided Design*, pp. 354-357, 1989.

[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. Int'l. Test Conference*, pp. 37-41, Cherry Hill NJ., 1979.

[Koe91] B. Koenemann, "LFSR-Coded Test Patterns for Scan Designs,", *Proc. European Test Conference*, pp.237-242, 1991.

[Kov79] P.G. Kovijanic, "Testability Analysis," *Proc. Int'l. Test Conference*, pp, 310-313, Cherry Hill NJ., 1979.

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

[Kra91] A. Krasniewski, A. Albicki, "Random Testability of Redundant Circuits," *Proc. ICCAD*, pp. 424-427, 1991.

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

[Kri86] B. Krishnamurthy, I.G. Tollis, "Improved Techniques for Estimating Signal Probabilities," *Proc. Int'l. Test Conference* , pp.244-251, 1986.

[Kri87] B. Krishnamurty, "A Dynamic Programming Approach to the Test Point Insertion Problem," *Proc. 24th Design Automation Conf.*, pp 695-704, 1987.

[Kri93] H. Kriplani, et al. "Resolvng Signal Correlations for Estimating Maximum Currents in CMOS Combinational Circuits," *Proc. Design Automation Conf*, pp. 384-388, 1993.

[Kub83] J.R. Kuban, W.C. Bruce, "The MC6804P2 Built-In Self-Test," *Proc. Int'l. Test Conference* , pp. 295-300, 1983.

[Lam95] D. Lambidonis et al., "A Quasi-Optimal Scheduling if Intermediate Signatures for Multiple Signature Analysis Compaction Testing Schemes," *Journal of Electronic Testing: Theory & Applications*, Vol.6, No. 1, pp. 75-84, February 1995.

[Lee90] S. Lee, K.G. Shin, "Design for Test Using Partail Parallel Scan," *IEEE Trans Computer-Aided Design* , pp.203-211, February 1990.

[Lin93] C-J Lin, Y. Zorian , S. Bawmik, "PBIST: A Partial-Scan Based Built-In Self-Test Scheme," *Proc. International Test Conference*, pp. 507-515, 1993.

[Lio89] A. Lioy, P.L. Montessoro, S. Gai, "A Complexity Analysis of Sequential ATPG," *Proc-ISCAS '89*, pp. 1946-1949, 1989.

[Ma 88] H. Ma et al., " Test generation for Sequential Circuits," *IEEE Transactions On CAD*, pp. 1081-1093, 1988.

[Maa90] F. Maamari, J. Rajski, "A Method of Fault Simulation Based on Stem Regions," *IEEE Trans. Computer Aided Design*, Vol. 9, pp 212-220, Feb. 1990.

[Man89] W.R. Mann, "R96MFX Test Strategy," *Proc. Int'l. Test Conference*, pp. 611-614, Washington D.C., Aug. 1989.

[Mal85] S. Mallela, S. Wu, " A Sequential Test Generation System," *Proc. Int'l. Test Conference*, pp 57-61, 1985.

[Mal91] S. Malik, et al., "Retiming and Resynthesis: Optimizating Sequential Networks With Combinational Techniques," *IEEE Transactions on CAD*, Vol. CAD-10, No. 1, pp. 74-84, Jan 1991.

[Mar78] R. Marlett, "EBT: A Comprehensive Test Generation System for Highly Sequential Circuits," *Proc. 15th Design Autoamtion Conf.*, pp. 332-338, 1978.

[Mat93] B. Matiew, D.G. Saab, "Augmented Partial Reset," *Proc. Int'l. Conf. Computer-Aided Design*, pp.716-719, 1993.

[Max91] P.C. Maxwell et al., "The Effect of Different Test Sets on Quality Level Prediction: When is 80% Better than 90%?" *Proc. Int'l. Test Conference*, pp.358-364, 1991.

[McC81] E.J. McCluskey, S. Bozorgui-Nesbat, "Design for Autonomous Test," *IEEE Trans. on Computers*, Vol. C-30, No. 11, pp. 860-875.

[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.

[McL92] G. McLeod, "BIST Techniques for ASIC Design," *Proc. Int'l. Test Conference*, pp. 496-505, 1992.

[Mei74] K.C.Y. Mei, "Bridging and Stuck--at Faults," *IEEE Trans. Computers*, Vol. C-23, pp. 720-727, July 1974.

[Moj93] M. Mojtahedi, W, Geissekhardt, "New Methods for Parallel Pattern Fast Fault Simulation for Synchronous Sequential Circuits," *Proc. Int'l. Conf. On Computer-Aided Design*, pp. 2-5, 1993.

[Moo56] E.F. Moore, *Automata Studies*, pp129-153, Princeton University Press, 1956.

[Moo92] J. Moondanos, J. Abraham, " Sequential Redundancy Identification Using Verification Techniques," *Proc. Int'l. Test Conference*, pp. 197-205, 1992.

[Mur90a] F. Muradali, V.K. Agarwal, B. Nadeau-Dostie, "A New Procedure for Weighted Random Buit-In Self-Test," *Proc. Int'l. Test Conference*, pp.660-669, Washington D.C., 1990.

[Mur90b] F. Muradali, V.K. Agarwal, Benoit Nadeau-Dostie, "Weighted Random Patterns for Built-In Self-Test," *BIST Workshop*, Kiawah Is, South Carolina, March 1990.

[Mur90c] F. Muradali, V.K. Agarwal, Benoit Nadeau-Dostie, "Weighted Random Patterns for Built-In Self-Test," *Testmethoden und Zuverlassigkeit vonx Schaltungen und Systemen*, Gut Ising, Germany, March 1990 *(invited paper.)*

[Mur95a] F. Muradali, T. Nishida, T. Shimizu,"A Structure and Technique for Pseudorandom-Based Testing of Sequential Circuits," *Journal of Electronic Testing: Theory & Applications*, Vol.6, No. 1, pp.107-115, February 1995.

[Mur95b] F. Muradali et. al., "A Register Structure for a Hardware Pseudorandom-Based Testing of Sequential Circuits" - *Japan Patent P406138188.*

[Mur96] F. Muradali, J. Rajski, " A Self-Driven Test Structure for Pseudorandom Testing of Non-Scan Sequential Circuits," *Proceedings 14th IEEE VLSI Test Symposium*, pp. 17-25, April 1996.

[Mut76] P. Muth, "A Nine Valued Circuit Model for Test Generation," *IEEE Transactions on Computers*, Vol. c-25, No. 6, pp. 630-636, June 1976.

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

[Naj92] F. Najm, "Transition Density: A New Measure of Activity in Digital Circuits," *IEEE Trans. CAD*, pp.310-323, 1992.

[Nar95] S. Narayanan, M.A. Breuer, "Reconfiguration Techniques for a Single Scan Chain," *IEEE Trans. on Computer-Aided Design*, Vol. 14, No. 6, pp. 750-765.

[Nie90] T.M. Niermann, W-T. Cheng, J.H. Patel, "PROOFS: A Fast, Memory Efficient Sequential Circuit Fault Simulator," *Proc. 27th Design Autoamtion Conf.*, pp. 535-540, 1990.

[Nie91] T. Nierman, J.H. Patel, "HITEC: A Test Generation Package for Sequential Circuits," *Proceedings European Design Automation Conference*, pp. 214-218, 1991.

[Ono91] T. Ono, M. Yoshida, "A Test Generation Method For Sequential Circuits Based on Maximum Utilization of Internal States," *Proc. Int'l. Test Conference*, pp 75-82, 1991.

[Par75] K.P. Parker, E.J. McCluskey, "Probabilistic Treatment of General Combinational Networks," *IEEE Trans. on Computers*, Vol. C-24, pp.668-670, June 1975.

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

[Par90] E-S. Park, M.R. Mercer, "An Efficient Delay Test Generation System for Combinational Logic Circuits," *Proc. 27th Design Auto. Conf.*, pp. 522-528, 1990

[Pat91] S. Pateras, J. Rajski, "Generation of Correlated Random Patterns for the Complete Testing of Synthesized Multi-Level Circuits," *Proc. Design Automation Conference*, pp. 347-352, 1991.

[Pom91] I. Pomeranz, S.M. Reddy, "On Achieving a Complete Fault Coverage for Sequential Machines using the Transition Fault Model," *Proc. 28th Design Autoamtion Conf.*, pp. 341-346, 1991.

[Pom91b] I. Pomeranz, S.M. Reddy, "Test Generation for Synchronous Sequential Circuits using Multiple Observation Times," *Proc. Fault Tolerant Comp. Symp.*, pp. 52-59, 1991.

[Pom94] I. Pomerantz, S. Reddy, "On Achieving Complete Fault Coverage for Sequential Machines," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 13, No. 3, March 1994.

[Pra91] D.K. Pradhan, S. Nori, J. Swamiathan, "A Methodolgy for Partia Scan Design," *Proc. European Test Conf.* , pp.263-270, 1991.

[Put71] G.R. Putzulo, J.P. Roth, "A Heuristic Algorithm for the Testing of Asynchronous Circuits," *IEEE Trans. Computers*, C-20, pp. 639-647, June 1971.

[Raj90] J. Rajski, H. Cox, "A Method to Calculate Necessary Assignments in Algorithmic Test Pattern Generation," *Proc. International Test Conference*, pp. 25-34, 1990.

[Raj93a] J. Rajski, J. Tyszer, "Test Response Compaction in Accumulators with Rotate Carry Adders," *Trans. on Computer-Aided Design*, pp. 531-539, April 1993.

[Raj93b] J. Rajski, J. Tyszer, "Accumulator-Based Compaction of Test Responses," *Trans. on Computers*, pp. 643-650, 1993.

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

[Rud92] E.M. Rudnick, V. Chikermane, J.H. Patel, "Probe Point Insertion for At-Speed Test," *Proc. IEEE VLSI Test Symp.*, pp. 223-228

[Saa92] Daniel G. Saab, Youssef G. Saab, Jacob Abraham, "CRIS : A Test Cultivation Program for Sequential VLSI Circuits", *IEEE International Conference on Computer Aided Design.*, 1992.

[Sab88] K. Sabnani, A. Dahbura, "A protocol Test Generation Procedure," *Computer Networks*, Vol 15, pp. 285-297.

[Sal88] K.K. Saluja, R. Sharma, C.R. Kime, " A Concurrent Testing Technique for Digital Circuits," *IEEE Trans. on Computer-Aided Design*, Vol. 7, No. 12, pp. 1250-1259, December 1988.

[Sav80] J. Savir, "Syndrome Testable Design of Combinational Circuits," *IEEE Trans. Computers*, Vol. C-29, pp. 422-451, June 1980.

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

[Sav89] Y. Savaria, et al., "A Pragmatic Approach to the Design of Self-Testing Circuits," *Proc. International Test Conference*, pp.745-754, 1989.

[Sav91] Y. Savaria et al., "Automatic Test Point Insertion for Pseudorandom Testing," *Proc. IEEE International Symp. Circuits Systems*, pp. 1960-1963, 1991.

[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.

[Sch89] M.Schultz et. al, "Essential: An Efficient Self-Learning Test Pattern Generation Algorithm for Sequential Circuits," *Proc. Int'l. Test Conference*, pp. 28-37, 1989.

[Sei91] B. Seiss, P. Trouborat, M. H. Schlutz, "Test Point Insertion for Scan-Based BIST," *European Test Conference*, pp. 253-262, 1991.

[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.

[Ses65] S. Seshu, "On an Improved Diagnosis Program," *IEEE Trans. on Electronic Computers*, Vol. EC-12, No. 2, pp. 76-79, Feb. 1965.

[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.

[Set85] S.C.Seth, L. Pan, V.D. Agrawal, "PREDICT - Probabilistic Estimation of Digital Circuit Testability," *Proc. FTCS-85*, pp.220-225, June 1985.

[Set86] S.C.Seth, V.D. Agrawal, "An Exact Analysis for Efficient Computation of Random Pattern Testability," *Proc. 16th FTCS* pp. 318-323, July 1986.

[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-PatternGeneration approach for VLSI Built-In Self--Test", Proc.ITC-88, pp. 256-262, Washington DC, Sept.1988.

[Sla92] S. Pilarski, A. Krasniewski, "Estimating Testing Effectiveness of The Circular Self-Test Path Techniques," *IEEE Trans. on Computer-Aided Design*, Vol. 11, No. 10, pp. 1301-1316, Oct. 1992.

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

[Sod89] J.M. Soden, et. al. "Iddq Testing - A Review," *Journal of Electronic Testing: Theory & Applications*, pp. 19-30, December 1992.

[Sou95a] M. Soufi, Y. savaria, B. Kaminska, " On Using Partial reset for Pseudorandom Testing," *Proc. Int'l. Symp. on Circuits and Systems*, pp. 949-952, 1995.

[Sou95b] M.Soufi, Y. Savaria, B. Kaminska, "On the Design of At-Speed Testable VLSI Circuits," *Proc. IEEE VLSI Test Symp.*, pp. 290-295, 1995.

[Ste76] J.E. Stephenson, J. Grason, "A testability Measure for Register Transfer Level Digital Circuits," *Proc. 6th Fault Tolerant Computing Symp.*, pp. 101-107, 1976

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

[Str95] A.. Stroele, H-J. Wunderlich, "Test Register Insertion with Minimm Hardware Cost," *Int'l Conference on Computer-Aided Design*, pp. 95-101, 1995.

[Sus73] A.K. Suskind, "Diagnostics for Logic Networks," *IEEE Spectrum*, Vol. 10, pp. 148-156, 1973.

[The89] K. Thearling, J.A. Abraham, "An Easily Computed Functional Level Testability Measure," *Proc. Int'l. Test Conference*, pp. 381-390, 1989.

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

[Tou96] N.Touba, E. McCluskey, "Test Point Insertion Based on Path Tracing," *14th VLSI Test Symposium*, pp. 2-8, 1996.

[Tri80] E. Trischler, "Incomplete Scan Path with an Automatic Test Generation Methodology," *Proc. Int'l. Test Conference*, pp. 153-162, 1980.

[Tui94] C-Y. Tsui et al., "Exact and Approximate Methods for Calculating Signal and Transition Probabilities in FSMs," *Proc. 31st Design Autmation Conf.*, pp.18-23, 1994.

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

[Ulr74] E.G. Ulrich, T. Baker, "Concurrent Simulation of Nearly Identical Digital Networks," *IEEE Computer*, Vol. 7, pp. 39-44, April 1974.

[Ven93] S. Venkataraman et al., "An Efficient BIST Scheme Based on Reseeding of Multiple Polynomial Linear Feedback Shift Registers," *Proc. Int'l. Conf. on Computer-Aided Design*, pp. 572-577, 1993

[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 et al., "Fault Simulation for Structured VLSI," *VLSI Systems Design*, Vol.6, No. 12, pp. 20-32, Dec. 1985.

[Wai85b] J.A. Waicukauski et al., " A Statistical Calculation of Fault Detection Probabilities by Fast Fault Simulation," *Proc. Int'l. Test Conference*, pp. 779-784, 1985.

[Wan86] L-T. Wang, E.J. McCluskey, "Concurrent Built-In Logic Block Observer (CBILBO), *Int'l. Symp. on Circuits and Systems*, pp. 1054-1057, 1986.

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

[Wun85] H-J. Wunderlich, "PROTEST: A Tool for Probabilistic Testability Analysis," *Proc. Design Automation Conf*, pp. 204-211, 1985.

[Wun88] H-J Wunderlich, "Multiple Distributions for Biased Random Test Patterns," *Proc. Int'l. Test Conference*, pp. 236-244,1988.

[Wun89a] H-J. Wunderlich, "The Design of Random-Testable Sequential Cirucits," *Proc. 19th International Symposium on Fault Tolerant Computing*, pp. 110-117, 1989.

[Wun89b] H-J. Wunderlich, S. Hellebrand, "The Pseudo-Exhaustive Test of Sequential Circuits," *Proc. International Test Conference*, pp. 19-27, 1989.

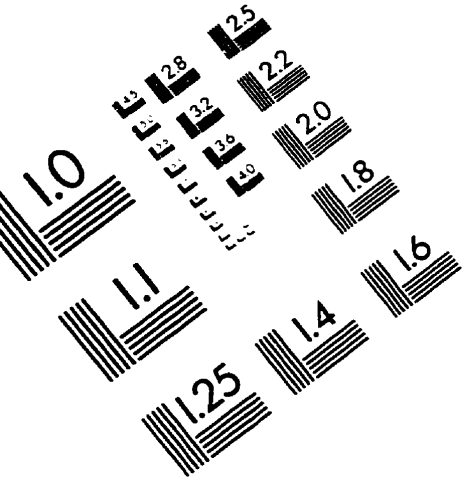
[Wu91] Eleanor Wu, "PEST: A Tool for Implementing Pseudo-Exhaustive Self-Test," *AT&T Technical Journal*, pp. 87-100 Jan./Feb. 1991.

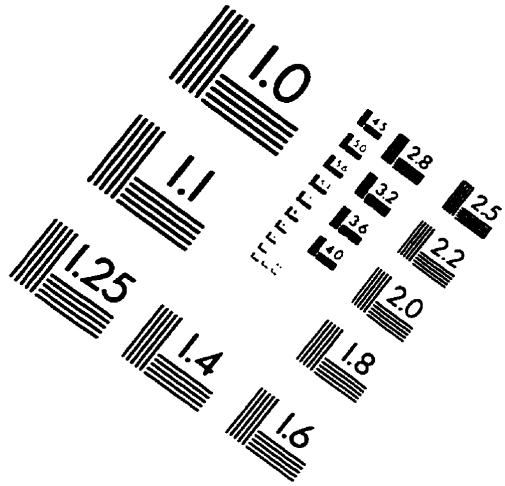
[Xav89] D. Xavier, R.C. Aitken, A. Ivanov, V.K. Agarwal, "Experiments on Aliasing in Signature Analysis Registers," *Proc. International Test Conference*, pp. 344-354, 1989.

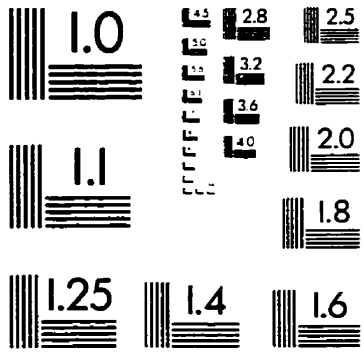
[Zac95] N. Zacharia, J. Rajski, J. Tyszer, "Decompression of Test Data Using Variable-Length Seed LFSRs," *Proc. IEEE VLSI Test Symposium*, pp. 426-433, 1995.

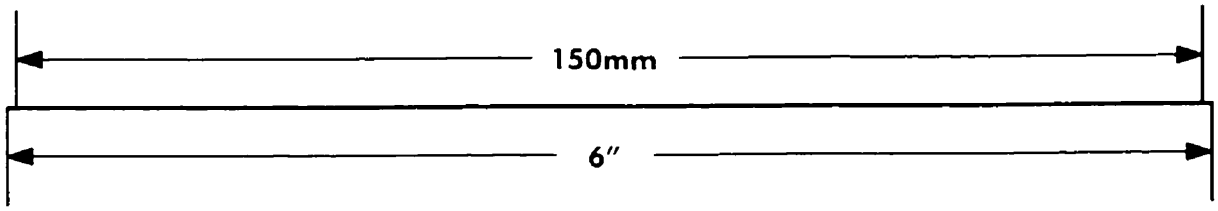
[Zor90] Y. Zorian, V.K. Agarwal, "Optimizing Error Masking in BIST by Output Data Modification," *Journal of Electronic Testing: Theory and Applications (JETTA)*, Vol. 1, pp. 59-71, May 1990.

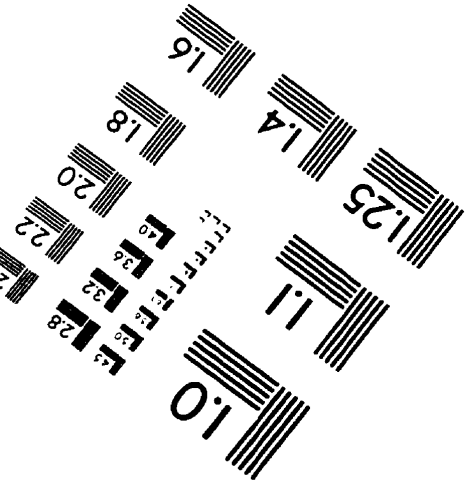
IMAGE EVALUATION
TEST TARGET (QA-3)

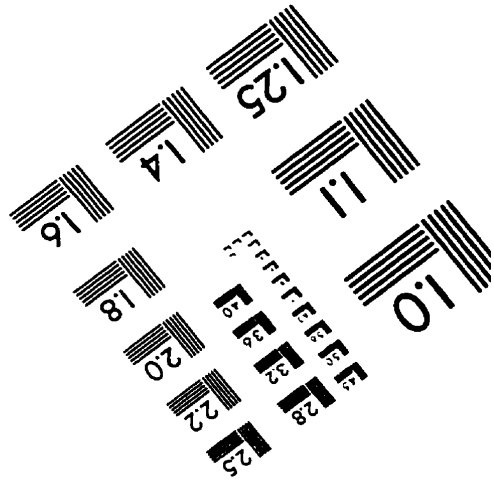












APPLIED IMAGE . Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989