INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

BIST Fault Diagnosis In Scan-Based Modules

Victor Zia

McGill University, Montreal

Department of Electrical Engineering

July 1998

A Thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Masters of Engineering.

Abstract

Testing digital devices constitutes a major portion of the cost and effort involved in their design, production, and use. Built-In Self-Test (BIST) has been warmly embraced by the Integrated Circuits (IC) industry as a solution for the continuously aggravating testing problem. BIST provides a simple go/no-go test screening answer. However, when test fall-out is high, it becomes necessary to diagnose faults to improve the yield.

Signature Analysis (SA) is typically used in a BIST environment to compact the outputs of a module into a final signature. Several SA-based diagnostic schemes have been developed in the past. An overwhelming majority of these techniques assume the presence of very few error bits in the Test Response Sequence (TRS). However, this assumption is generally unrealistic since a faulty device in a practical BIST environment can generate an enormous number of erroneous bits in the TRS.

In this thesis, a comprehensive survey of the current SA-based BIST diagnostic schemes is presented first. Then, novel BIST fault diagnosis techniques for scan-based VLSI modules are presented, based on multiple signature analysis.

The proposed schemes do not assume any specific fault model and thus, can be used to diagnose all voltage-detectable faults. Diagnosing various malfunctions in a module begins by identifying the faulty scan elements that capture errors during test. Based on multiple faulty-signature information, the schemes guarantee the identification of these cells, regardless of the number of errors the module may produce in the TRS.

The techniques can be used at all levels of integration to provide chip level, printed circuit board, multi-chip module, and system level diagnostics. At the chip level, the diagnostic results are combined with well-developed classical diagnostic techniques to provide gate level or even transistor level diagnosis.

Two diagnostic algorithms have been developed, to be used with the proposed Built-In

Self-Diagnosis (BISD) module. An *adaptive* diagnostic algorithm is developed to minimize the number of signatures that must be collected during diagnosis. The algorithm does not impose any limit on the number of faulty scan elements, and can exactly identify any number of them.

Another *non-adaptive* diagnostic algorithm has also been developed. In this algorithm, an upper limit on the number of faulty scan elements is assumed. The identification of the faulty elements is guaranteed whether the actual number of these cells is greater than, equal to, or less than the set upper limit. However, the identification *may* not be exact, where the algorithm may declare non error-capturing scan cells as faulty cells.

Software simulations, assuming various modules with different scan chain sizes and different number of error-capturing scan elements, are used to quantify and compare the performance of the above algorithms. The experimental results support the use of the new BISD technique as a practical solution to the challenging diagnosis problem in scan-based BIST environments.

Résumé

Le test de circuits intégrés numériques représente une part importante des coûts et des efforts associés à leur développement, production et mise en application. Le procédé d'auto-test in situ ou Built-In Self-Test (BIST) a donc été accueilli avec enthousiasme par l'industrie des semi-conducteurs comme une réponse efficace au problème grandissant du test des circuits intégrés. BIST rend un simple verdict de type go/no-go. Cependant, un fort taux d'erreurs rend indispensables des moyens de diagnostic plus poussés, afin de localiser la source du défaut.

Dans ce but, l'analyse de signature (Signature Analysis - SA) est couramment utilisée conjointement au test BIST. Plusieurs techniques basées sur la SA ont été développées dans le passé. La plus grande majorité d'entre elles repose sur l'hypothèse d'un très faible nombre de bits erronés dans la séquence de réponse (Test Response Sequence -TRS). En pratique, cette hypothèse est rarement vérifiée dans la mesure où un circuit intégré défectueux peut générer un très grand nombre d'erreurs dans la TRS.

Cette thèse présente dans un premier temps un tour d'horizon détaillé des techniques de diagnostic BIST basées sur l'analyse de signature. De nouveaux procédés BIST basées sur l'analyse de multiples signatures et destinés au diagnostic de modules VLSI sont ensuite proposés.

Ces nouvelles techniques présentées ici ne sont basées sur aucun modèle d'erreur et par conséquent peuvent être appliquées au diagnostic de toute erreur décelable par mesure de tension. La recherche des causes du dysfonctionnement d'un module nécessite dans un premier temps l'indentification des cellules ayant capturé des erreurs lors du test. En s'appuyant sur l'analyse de plusieurs signatures, les techniques proposées garantissent l'identification de ces cellules, indépendamment du nombre d'erreurs générées dans la TRS.

Ces techniques s'appliquent à différents niveaux de diagnostic: du composant à la carte

électronique jusqu'au système. Appliquées aux circuits intégrés, les résultats obtenus par ces procédés peuvent être associès à des méthodes de diagnostic classiques pour localiser la source de l'erreur au niveau de la porte logique, voire même du transistor.

Deux algorithmes associès à la méthode BISD (Built-In Self-Diagnosis) ont été développés. Un premier algorithme *adaptatif* permet de minimiser le nombre de signatures nécessaires au diagnostic. Cet algorithme n'impose aucune limite dans ses hypothèses quant au nombre maximum de cellules capturant une erreur. De plus il garantit l'identification de ces dernières quelque soit leur nombre.

Un second algorithme, *non-adaptif* fait l'hypothèse d'un nombre maximum de cellules capturant une erreur. L'identification des éléments représentant une erreur est garantit que leur nombre soit supérieur, égal ou inférieur à cette limite supérieure. Toutefois, il se peut que l'algorithme déclare une cellule "saine" comme une cellule ayant une erreur.

Les performances de ces algorithmes ont été simulées et comparées en fonction de différent types de modules, de "scan chain" de différentes longueurs et du nombre d'éléments capturant des erreurs. Les résultats expérimentaux démontrent l'efficacité de la méthode BISD proposée et plaident en faveur de son utilisation dans le diagnostic de circuits intégrés numériques.

Acknowledgements

First and foremost, I would like to take this opportunity to express my gratitude to my thesis supervisor, Dr. Janusz Rajski, for accepting me into the Masters program and for his guidance during the term of my study. I am indebted to him for his insight, knowledge, and enthusiasm. There have been many disappointing times during this project, and I thank Dr. Rajski for encouraging me to explore yet another approach, finally leading to the techniques encompassing this work. The knowledge and thinking/reasoning process that I acquired with him through our discussions, will undoubtedly positively mark my future work, and hence life. For all of that, I am most grateful.

I am also greatly indebted to my co-supervisor, Dr. Jerzy Tyszer, for his invaluable advice and instructive guidance concerning this project and all the other personal and technical discussions. His support and research directions were instrumental in formulating the second technique presented in the thesis.

I wish to acknowledge the help of the VLSI support staff, in particular that of Jacek Slaboszewicz, the system manager, and Alain Magloire.

Due thanks must go to my friends in the VLSI Design Lab, present and past members, for their assistance and for making my stay here a truly enjoyable experience. I would especially like to thank Aiman, Nagesh, Nilanjan, Nadime, Fidel, Myriam, Victor, Michael, Eric, Arman and Mourad. Special thanks to Laurent Ronc for his assistance in writing the French version of the abstract.

Finally, I am enormously grateful to my family for their patience, unqualified love, constant encouragement, understanding and support in my many years of study. Special thanks to my sisters Vivian and Lilian for their support and help.

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

Dedicated

To my family

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Testing digital devices constitutes a major portion of the cost and effort involved in their design, production, and use [1]. Generating practical and effective tests for today's complex circuits has emerged as a major problem [9]. The testing problem has been continuously aggravated by the increasing levels of silicon integration. In fact, experts consider the testing problem to be a major obstacle in preventing the full exploitation of new technologies such as Multi-Chip Modules (MCMs) [10]. Consequently, the test technology has to evolve to meet the testing requirements of these devices.

Integrated circuits are produced by manufacturing a wafer of silicon having many copies (called die) of the IC. Typically, only a fraction of the die work properly [11]. Thus, the wafer must be tested in a process called *wafer-sort*, to sort out the working die. The die that pass wafer-sort are packaged.

Whether or not the chip carries out the function it was intended for is also verified in a process called *functional* testing. Traditionally, ICs have been tested by applying functional test vectors on high speed testers [12]. The test vectors attempt to exercise all the logic in the chip by causing it to execute all of its functions. Failures are modeled at the Register Transfer Level (RTL) or functional level in terms of the deviations in the expected function. A functional test must also verify that no unintended function has been additionally performed.

Although functional testing can realistically account for the actual effects of physical failures on the logic [1], it is extremely difficult to isolate the failures. Another problem with this approach is that an exhaustive set of functional test vectors is usually impractically

large, such that only a subset of it is used in testing. As a result, one never knows of the existence of physical defects that will result in a fault for some test vectors which are not included in the test set. These problems with functional testing drove most of the work on testing to the structural domain [1].

In *structural* testing, the Circuit Under Test (CUT) is first described in terms of logic primitives. Each of the primitives are faulted in turn, and a test is generated for each fault. In a complex CUT, many different failure models are possible. The assumption that faults can be modeled by logic gate inputs or outputs stuck to either a logical 1 or logical 0 values [3], has been widely used despite the fact that it does not cover all the possible failure modes.

1.1 Design For Test and BIST

It is common knowledge that designers are capable of placing a large number of components on a single chip by using current Very-Large Scale Integration (VLSI) and Ultra-Large Scale Integration (ULSI) technologies. As VLSI devices increased in complexity in recent years, the difficulty and cost of testing and failure analysis have also increased rapidly in commensurate proportions.

One obstacle to testing such circuits is the excessive volume of test data that must be stored. The Automatic Test Equipment (ATE) has become very expensive, and the time required to develop and apply high fault coverage tests has become prohibitively long. This is mainly due to the fact that the ratio of devices on a chip to the number of I/O pins is constantly increasing. Validating the state of a digital module has become all but impossible, without direct access to its internal state.

Having said that, it appears that the only practical solution to easing the test problem and reducing the test cost, lies in designing the testability into the circuits themselves. The term Design For Testability (DFT) describes a set of design techniques that make the process of product testing more economical. Most of these techniques increase the observability and controllability of a product's internal state. The higher the degree of test access, the simpler the process of test generation will be [12].

Scan design is the most widely used DFT technique in modern VLSI modules. A scan design in *test-mode* connects some or all of the state elements in a module into a single or multiple chain(s), through which test vectors can be applied. This allows the test pattern

generation tools to develop tests for strictly combinational circuits, rather than the much more complex sequential ones.

Because of the high levels of silicon integration, it would also seem reasonable that a small portion of the circuits in a module be devoted to the testing of the remaining ones. This concept is termed BIST.

BIST has been warmly embraced by the IC industry [1, 13, 14], as a practical solution for testing today's large and complex VLSI modules and tomorrow's even larger and more complex ULSI modules in deep sub-micron [15, 16].

In BIST, the test pattern generation and the output response analysis are both conducted on the same module as the CUT. In order to reduce the large amount of data space required to store the responses of the CUT, the responses are compacted into a small *signature* of a few bits which capture the effects of any existing faults. Although several compactors have been used in practice, *Signature Analysis* (SA) has been by far the most widely used data compaction technique [1].

At the end of a test session, the resulting signature is compared to the expected one from a fault-free circuit. A fault is detected if the two signatures are different.

BIST has been shown to be more effective than classical external testing. Since most of the test can be performed by circuitry resident on the module itself, the test equipment requirements for the CUT can be greatly simplified, and the dependence on the latest, most sophisticated external test systems is greatly reduced. This results in a reduction of the total test effort for that CUT which in turn considerably decreases the overall testing cost.

Ideally, solving the testing problem may require that every chip incorporates both DFT and BIST [9]. Incorporating DFT and BIST into an IC enables a high level of test quality and a reduction in the IC test time and cost. Understanding their economics is essential in determining the proper amount of DFT and BIST to include at the different module levels.

The analysis of the trade-offs associated with test strategies for complex modules clearly indicates that incorporating DFT and BIST with varying degrees at the chip, board or MCM level is economically justifiable [9, 17].

1.2 Diagnosis Basics

In addition to fault detection, an IC manufacturing test facility must also provide some way of identifying what went wrong [18]. This process is called *fault diagnosis*.

Diagnosis is the process of determining the cause(s) of circuit failure once they have been identified to be faulty by a detection test set. In other words, diagnosis maps the observed misbehavior of the CUT into physical faults affecting its components or their interconnections [3].

Diagnosing a CUT requires the determination of a set of diagnostic test patterns which, when applied to the CUT in question, detects and *locates* all the modeled faults or their combinations which are consistent with the observed symptoms. The degree of diagnostic resolution must be sufficient to define the necessary corrective action [19].

Fault location is often more difficult than fault detection and usually requires the application of more test vectors [20]. This is mainly due to the large number of failure sources and the need to accurately isolate such failures [21].

1.3 The Critical Role of Fault Diagnosis

As the IC industry evolves towards the 21st century, many trends are developing, making the diagnosis of VLSI modules a formidable challenge.

Competition and product innovations have been pushing IC manufacturing towards building more circuitry into smaller packages. Device and system operating speeds are growing faster than our ability to either test or verify [16].

Process technologies have kept in step by continuing to advance both horizontally, with reduced gate dimensions approaching the theoretical limits, and vertically, with no limit in sight to the number of interconnect layers [16]. These changes are resulting in continuously smaller and more complex products.

Product manufacturing and testing goals, however, have not changed: higher quality, reduced cost, shorter time to market, and correct functionality [22]. Diagnosis plays a critical role in achieving these goals as explained next.

1.3.1 Higher Quality and Reduced Cost

Customer expectations are for a higher level of product quality and reduced product cost. In order to keep current customers and gain market share, products must be provided with correct functionality, at the right cost, without compromising quality [22]. Nowadays, cost and quality have achieved the same level of importance as functionality [16].

To produce cost-effective, high-quality products, testing and fault diagnosis must be included as critical requirements early in the design cycle [23]. Treating them as an afterthought can result in higher costs and less reliable products [9].

Thus, the role of failure analysis has changed from that of a reactive one to that of a pro-active one [24]. Waiting for failures to come back from the field, in order to assess the reliability of a product, is not a valid option. By that time it is too late, and customers will find alternate supply sources [24].

It is the role of the pro-active failure analysis to conduct diagnostic testing in order to assist the designer in eliminating the cause before the parts ship. This results in a high-quality, cost-effective and more reliable product in the field [24].

1.3.2 Shorter Time to Market

Competition is driving the need for a fast Time To Market (TTM). New products are being developed more quickly. Design cycle times are decreasing inversely proportional to the product densities and operational speeds [16]. The test development time has to keep in pace.

In today's world of fast product introductions, it has become essential and often critical to have high testing throughput and fast, accurate diagnostics [22, 24]. Quickly debugging the root cause of failures and implementing the corrective actions, results in significant time and cost savings [25].

Time to market will differentiate the players in this highly competitive IC industry [16]. Diagnosis in conjunction with well designed parts, where testability and diagnosability are taken into consideration , will make a fast-turn TTM possible [24].

1.3.3 System-Level Diagnosis

Fault diagnosis is important at all levels of silicon integration. First at the system level, if a fault is detected, then the system cannot be simply discarded since it is the most expensive part.

There are several system diagnostic goals. The simplest is diagnosis for field repair, where the primary objective is to minimize the test time and improve the maintenance throughput, thus increasing the availability of systems. This is achieved by identifying and isolating the faults to system modules rather than the individual failure sources within a module.

Locating the faulty board(s) or multi-chip module(s) in the system is the first step in repairing it. Once the replaceable faulty modules are located, they can be replaced in the field by fault-free ones and the system *down time* can be minimized.

System level diagnosis is also an integral part of the process of achieving fault-tolerance [26]. Fault tolerance is defined as the ability of systems to operate correctly in the presence of faults or malfunctions [27]. A fault tolerant system includes idle spare components in addition to the ones in operation.

To achieve fault tolerance, the system is tested first for fault detection. Once a fault is detected, fault diagnosis is used to locate the faulty component(s). Depending on the diagnosis results, the system must reorganize the hardware to replace the faulty components by spares. Finally, a fault recovery process is initiated to restore a system to its correct state, so that it can continue to operate without its normal behavior being disrupted [28, 29].

Two trends have incited interest in fault tolerant systems. The need for extremely reliable digital systems is one major trend. Society has become increasingly dependent on digital systems for applications in which mistakes can be costly, devastating, or even life-threatening. Consequently, fault-tolerance has become a basic requirement in these systems [27].

A telecommunication system is one such system. Telecommunication equipment is often located in remote areas where manual service is not available all the time. Remote failure diagnosis is essential in achieving tolerance in such systems [30].

A second trend is the feasibility of constructing multiprocessor systems. Due to the advancing VLSI technologies, it is economically feasible to fabricate large array processor

systems of identical cells interconnected by a regular structure [29]. These systems can be used to attain high throughput, and are of great interest in application areas such as image processing, and matrix computations. Their regularity which makes them suitable for VLSI implementations, also tends to make them error prone [31]. Locating the faulty processors is again a prerequisite for reconfiguration and fault tolerance [32, 33].

1.3.4 Board-Level and MCM Diagnosis

Moving down to the next level, if a fault is detected in a board or MCM, then it also cannot be simply discarded. The reason is again economics.

The growing complexity of various life applications has resulted in subsystems built around boards consisting of a large number of chips [34]. Boards are produced in low volumes as compared to the production volumes of the chips mounted on them. A multi-layered double-sided board, tightly packed on both sides with Surface Mount Technology (SMT) components, could cost several thousand dollars [35]. Hence, circuit boards cannot be just discarded, since the cost of a new one is higher than the cost of repairing it.

In addition, a board may contain several *expensive* chips. These chips include some highly sophisticated VLSI and Application-Specific Integrated-Circuit (ASIC) chips. These chips, such as the ones used in military applications, are usually manufactured in small volumes. Thus, once they are mounted on the board, a board can not be simply sacrificed just because a single or a few other inexpensive chips on the board are faulty.

Again, locating the faulty chip(s) on the board rather than the failing logic inside the chip(s) is a prerequisite for efficient repair [18]. The problem of diagnosing such complex boards has become a major bottleneck [34].

Once the faulty chips are located, they can be replaced, and the board can be reinstalled in a system again.

Similar to a board design, MCM technology allows placing many ICs on a common substrate in a single package, for increased reliability and decreased size and weight [12]. Either seen as complex integrated circuits or as compact boards, MCMs have become one of the most suitable answers to today's application needs, be it from avionics, consumer electronics, automotive or telecommunications. The use of MCMs is projected to dominate the electronics industry in the 21st century [36].

As the MCM technology becomes more sophisticated, the number of expensive chips

mounted on an MCM grows rapidly. As a result, the need for repairing the faulty MCMs is becoming more and more important, where replacing a bad chip is far more cost effective than throwing away a whole MCM module [37].

1.3.5 Chip-Level Diagnosis

Finally, moving down to the chip level, if a fault is detected, then it is again important to locate the source(s) of failure(s) inside the chip.

Since a defect in a chip is often not repairable, and since chips are produced in large volumes, it is usually economical to just discard the faulty ones. The naturally rising question is, then why diagnose faults in chips that are going to be discarded anyway?

First, if a fault is found that explains the failure data collected during test, then design errors can be discovered by inspecting the design databases. Discovering these errors is the first step in correcting the design in a future silicon stepping.

Failure diagnosis is also critical to the chip manufacturer. Ideally, when all the manufacturing processes operate properly, the end product is good. However, due to the intricate nature of IC manufacturing, even the best manufacturing processes sometimes produce ICs with defects [22].

During the start-up phase of production, identifying the source(s) of chip failure(s) is an efficient way of identifying process errors for the purpose of yield improvements. Both zero yield and low yield are conditions that signal the need for failure diagnosis [18].

The fabrication errors can be discovered by inspecting the fabrication databases or the faulty device itself with the aid of an electronic-scanning microscope. Using the diagnosis results, the settings and the control of a manufacturing process can then be adjusted.

Finally, fault diagnosis is also important in improving chip quality. Diagnosing a chip that passes the original production test and then fails in the field, can provide information on the design and/or fabrication weaknesses. Samples of faulty chips that exhibit common recurring failure modes are good candidates for such diagnosis. Utilizing the diagnosis results, the design and/or fabrication can be improved to minimize future occurrences of the failure [38].

To summarize, in order to remain competitive in the marketplace, IC designers and manufacturers must be able to rapidly debug new products and processes and usher them

quickly to high volume production [39]. This leads to high-quality chips being manufactured, which in turn decreases the diagnostic effort required at the higher MCM, board, and system levels.

1.4 Classical Diagnosis Methods

Traditionally, fault simulation methods have been used to provide an automated diagnosis, using fault dictionaries.

A fault dictionary is a cross-reference list that associates each fault in the list with the measures defined for each test vector in the diagnostic test set [18]. Each erroneous bit at the output is called a failing measure for that particular test.

The dictionary database is created by simulating a reference copy of the circuit for each of the modeled faults in the list. A fault-dictionary generation program uses the simulation outputs, to compile lists of failing-measures/pattern combinations which uniquely describe the simulated faults.

During testing, an external tester applies the test patterns, and stores the observed CUT outputs in a data-log. If a module fails during test, a lookup program examines the tester failure data-log and matches the failing measures to faults in the dictionary. Close matches indicate the likeliest fault source(s) [40].

In addition to fault dictionaries, various other Failure Analysis (FA) processes can be used to determine the correct root cause of failures. Powerful FA processes continue to be developed, based on internal IC scanning using an electron beam, ion beam, photon emission microscopy, optical microscopy (visible, infrared, or scanning laser microscopy), thermal infrared scanning, scanning probe, and other techniques combined with other innovative test methods [41, 42]. Physical access to a defect site is also possible with wet or dry etching for layer removal and cross sectioning.

These FA techniques can also be combined with the fault dictionary method to obtain more efficient IC diagnostics. The faults identified by the dictionary method can significantly narrow the search space for subsequent FA techniques.

Moving up to the next level, in-circuit testing in conjunction with functional testing are traditionally used to test and diagnose boards.

In-circuit testing, is a test practice in which the mounted chips are tested one at a

time [12]. In order to test each chip on the board, the tester must have direct physical access to the individual chips, board edge connectors and vias. The chip access method that has been used for many years is a bed-of-nails test fixture contacting each pin with a spring-loaded probe. Since each chip is tested separately, diagnosis is readily achieved.

The other traditional method of testing a board has been with a functional tester through an edge-connector, or a test bus such as the boundary scan bus [43, 44]. Diagnosis is typically achieved through a combination of custom diagnostic routines, guided probing from a failed board output back to the chip node causing the failure, and the use of electronic instrumentation such as a logic analyzer or an oscilloscope [45].

In some applications, a fault dictionary is used in place of, or in addition to, the guided probe [40]. By matching the failed responses of a board with a similar set of responses in the fault dictionary, it is possible to map back to the likeliest failure locations.

Next, at the MCM level, a module can be tested and diagnosed by treating it as either a very compact board [12] or a very complex chip.

Treating an MCM as a compact board, fault diagnosis is achieved with a tester and a fault dictionary or guided probe [12]. Treating it as a complex chip, electron-beam probing is the likely FA tool [46].

Finally, at the system level, diagnosis is achieved by extensive diagnostic software routines exercising the normal functions of the system in an attempt to isolate the occurring hardware failures [47].

1.5 Inadequacies of Classical Diagnostic Methods

The aforementioned classical diagnosis methods are facing formidable challenges in the VLSI era.

First at the chip level, VLSI circuits fabricated in increasingly deep sub-micron technologies are making the job of diagnosing failures more difficult [15]. Traditional methods of failure analysis are becoming inefficient and less effective.

New diagnostic capabilities are required for VLSI ICs with new features such as larger chip area, finer chip dimensions, higher speeds, additional interconnection layers, power distribution planes, and flip chip or array bonding packaging configurations [42]. These

features, reduce or prevent the use of conventional FA methods, and make it more difficult to recreate and observe failure conditions, without destructive de-processing [15, 42].

Fault isolation with a fault dictionary, requires an excessive computational time and space for large chips [23]. Even moderately sized circuits can require an enormous amount of test stimulus to ensure a sufficient *diagnostic coverage* [19]. Not withstanding the recent advances in speeding up the simulation algorithms [48, 49], diagnostic fault simulation remains a computationally difficult problem [50].

Traditional, labor-intensive FA techniques such as liquid crystal analysis or photon emission microscopy, while still quite valuable, are beginning to become less useful [39]. Infrared optical techniques are limited in the defect mechanisms they can find [25].

The packaging technology is bringing into prominence other limitations that may be placed on FA tools by the physical structure of the circuit. Internal probing is often ineffective in VLSI chips, because there are normally no probe points except at the device I/O pads. Even the newer non-contact electron-beam probing has limited use. This technique is not effective when there are many layers of metal. The metal and/or silicon can be removed to provide internal circuit access, but that might destroy other parts of the chip. Moreover, this technique is only useful if the defects have already been localized to a small area [18]. Choosing the appropriate probe points continues to be a serious bottleneck [51].

Moving up to the next level, the increasing levels of integration, advances in the physical interconnect technology and more advanced packing technologies are all challenging the diagnostic capabilities of traditional, board-level FA tools.

Today's VLSI boards typically include large pin-count devices, multiple internal and external data buses, ASICs, SMT devices, multi-layer board structures, MCMs, and components with very high speeds [40].

For decades the dominant IC packaging technology has been the Dual-In-Line (DIL) package with 100 millimeter pin spacing, mounted through holes in a Printed Circuit Board (PCB) [35]. However to meet the increasing speeds and packaging density requirements of modern VLSI chips, the circuit board topology has shifted in recent years towards leadless chip carriers, SMT, and Tape Automated Bonding (TAB) as alternatives to through-hole mounting [22].

SMT for example, permits physical compaction by using closer pin spacing of 25 millimeter or less, and by mounting components on both sides of the board [35, 45]. This has limited the applicability of in-circuit testing, because direct tester access now requires an

expensive double-sided bed-of-nails test fixture, adding unacceptable tester costs. Conformal coating on a board may rule out in-circuit testing all together [52].

Because of the aforementioned physical access restrictions, traditional board-level functional testing methods have begun to break down [45]. The VLSI functional testers are costly and provide inadequate fault diagnosis support [12]. Complex boards require high-quality software diagnostic routines (test programs) to aid fault isolation. The quality of a test program might be impeded by the lack of models, component wiring configurations, and the tester access to a board's node [22].

Without physical access, guided probe methodologies are no longer feasible unless test points (pads) are added [45]. Modern boards are becoming so dense that separate test pads for probing are considered to be a limiting factor in space utilization that negates the packing density advantage of SMT [12, 35, 52]. In addition, the growing complexity of modern VLSI boards slows guided probing by multiplying the number of points that must be contacted, and requires a large database [40].

The non-invasive fault dictionary method works independently of a board package, but the increasing levels of integration and potential fault sites are making it harder to provide a good diagnostic resolution [40]. A larger fault universe requires an extremely efficient diagnostic fault simulator, to create a dictionary large enough to be useful [40].

Next, at the MCM level, either seen as a compact board or as a very complex chip, an MCM suffers from the aforementioned chip and board problems.

There are no commercial MCM test fixtures in widespread use [12]. The MCM-level functional test development is an astronomical challenge [53]. Since an MCM is usually faster than the delay along the tester probe circuitry, diagnostic probing is out [9]. Finally, MCMs are much too complex to generate a fault dictionary [12].

Finally, at the system level, the effort spent for extensive diagnostic packages is huge, and must be diminished [47].

1.6 Thesis Motivation

At this point, it must be clear that the classical test and diagnosis methods are both ineffective and inefficient for complex VLSI modules. However, as mentioned earlier, BIST when combined with efficient DFT techniques, can substantially ease the aforementioned

testing problems.

Fault diagnosis methodologies are tightly coupled with the test methodologies [38]. In SA-based testing, the only available failure data is the final signature. By comparing it against the pre-computed reference signature, failures can be easily detected.

Although test response compaction with SA has significantly eased the problem of Output Response Analysis (ORA), it has added more difficulties to fault diagnosis. A faulty signature is of little use in diagnosis. This is because the individual test responses and their associated failing measures are lost by the compaction process. Without these failing measures, fault diagnosis is simply impossible.

Hence, we are faced with a new challenge in BIST, namely, the diagnosis problem. Since we cannot determine whether the individual responses are faulty or not, the complete failure data (all test patterns and responses) must be retained during diagnosis. This, however, implies a huge volume of data that must be collected and stored, which is not practical, and negates the cost savings and benefits of on-module test pattern generation and ORA.

Hence, to diagnose a fault in a BIST environment, it is first necessary to decode the faulty signature to identify the test vectors and the corresponding module outputs that respectively detect and capture errors during test. Once done, the traditional non-BIST diagnostic techniques can be used to obtain a higher resolution diagnostics.

Several research efforts have been devoted to methods for identifying the failing tests and/or faulty components that cause the faulty signature. This is the subject of the first part of the thesis, where a comprehensive survey is presented of the diagnostic techniques that can be used with SA-based BIST structures at the system, board, MCM, and chip levels.

An overwhelming majority of these techniques assume either the presence of very few error bits in the TRS, or very few error capturing outputs. However, this assumption is generally unrealistic since a faulty module in a practical BIST environment can generate an enormous number of erroneous bits in the TRS, and can contaminate many outputs. Hence, these techniques currently offer no practical solution to the VLSI diagnosis problem in BIST environments.

More work is necessary to ease the diagnosis of failures in SA-based BIST environments [15]. New paradigms for failure analysis are needed to meet the challenges posed by ICs that have more than 10^6 transistors with feature sizes less than 0.2 micron, operating over 250 MHz [42]. There is a critical need not only for a rapid evolution of existing

diagnostic and FA capabilities, but also for breakthroughs, enabling rapid failure isolation [42]. Automated BIST diagnostics in modules designed with an aggressive approach to DFT, will become the dominant strategy optimizing the time to market of digital products [16, 39].

Motivated by the above, novel BIST fault diagnosis techniques are presented for a specific type of testable modules, namely, the scan-based modules. Fault diagnosis of non-scan modules is a nontrivial problem which has not been studied in any depth. Thus, most diagnosis techniques are targeted for scan modules [54].

Diagnosing various malfunctions in a module begins by identifying the scan chains and scan cells (flip-flop or latch state variables) that capture errors during test. These scan chains/cells will be referred to as the discrepant scan elements [55] or faulty scan elements. Based on multiple SA information, our schemes guarantee the identification of these scan cells, regardless of the number of errors the module may produce during test. These techniques can be used at all levels of integration to provide chip, MCM, board and system level diagnostics. The diagnosis process is *hierarchical* such that a faulty unit identified at one level becomes the Unit Under Test (UUT) at the next lower level. The proposed schemes do not assume any specific fault model and thus, they can diagnose all voltage-detectable faults.

As will be explained later, locating the discrepant scan chains is sufficient for system, board, and MCM level diagnostics. At the chip level, locating the discrepant scan cells is very helpful in diagnosis since most of today's VLSI chips are designed at the RTL level [38], and are automatically synthesized by design tools [56]. This means that a designer or a test engineer is more familiar with the scan cells than the inner circuits of such modules [56]. Without knowing where to look, the diagnosis process cannot continue and there are no alternatives [25]. Identifying the faulty cells provides coarse localization of failure sites. This information can be fed to a FA tool to provide automated gate level or even transistor level diagnostics.

Two diagnostic algorithms are developed to be used with our scan-based BISD architecture (GSTUMPS). An *adaptive* algorithm is developed to minimize the number of signatures that must be collected during diagnosis. Another *non-adaptive* algorithm is also developed to minimize the tester socket time.

The use of our BISD architecture results in highly reliable digital modules since faults can be located immediately after detection, and a recovery process or a corrective action

can be executed immediately.

1.7 Organization of The Thesis

The thesis is organized as follows: In Chapter 2, the DFT and BIST techniques are summarized. Several SA conventions are developed in that chapter and are followed throughout the thesis. Next, the chip-level BIST diagnosis is discussed in Chapter 3. Several SA-based diagnostic techniques are discussed, outlining their applicability, practicality, diagnostic efficiency and cost. Moving up to the remaining levels, MCM, board, and system level BIST diagnostics are explained in Chapter 4. Again, several SA-based diagnostic procedures are reported. The scan-based BISD architecture (GSTUMPS), to be used with our diagnostic algorithms, is presented in Chapter 5. Also in that chapter, analytical signature relations are developed to be used with these algorithms. In Chapter 6, the adaptive diagnosis algorithm is presented. The non-adaptive algorithm is presented in Chapter 7. Based on our diagnostic methods, a conceptual failure analysis system is developed in Chapter 8 to provide gate level and transistor level diagnostics. The thesis concludes with Chapter 9.

Chapter 2

DFT & BIST

An obvious approach to alleviating the need for sophisticated VLSI testers is to incorporate the tester functionality into the MUT itself, at all levels of integration. Hence the notions of DFT and BIST. Incorporating DFT and BIST into a digital module eliminates the need for expensive testers and provides a mechanism for accessing and exercising the internal module circuitry at its normal speed. DFT and BIST, however, are not free. Understanding their economics is essential in determining the proper amount to include at each module level.

In this chapter, a brief survey is presented of the most popular DFT and BIST techniques used in modern VLSI designs.

2.1 Design For Testability Techniques

During the seventies, a quiet revolution occurred in the field of digital module testing. Before that time, testing was a manufacturing core almost completely isolated from design. After that time, the burden of test was shifted to the designers [1].

With the advent of VLSI technologies in the following decade, circuit geometries became smaller and more compact, resulting in an exponential increase in the circuit densities available in a single package.

The increase in packing density, which caused dramatic savings in circuit costs, also caused testing to consume a higher percentage of the these costs. This effect was forced

primarily by the increase in the gate-to-pin ratio, making the observation and control of internal circuit nodes more difficult.

In 1975, Phillip Writer coined the term "Design for Testability" [57] advocating a set of design styles meant to improve the observation and control of the designs during test. By 1980, the need to design for easier testing had taken hold.

Testability depends heavily upon the control the designer has over his/her implementations. There are certain properties of a design that makes it easier to test. Such a design contains no logical redundancy, it contains no asynchronous logic, its clocks are isolated from the logic, its sequential circuits are easily initialized, it may include test points, and it has a minimal area or pin overhead over a *normal* design.

These techniques, collectively referred to as ad-hoc DFT techniques, are just a collection of good design methods based on practical experiences of conventional test generation for non-DFT designs. They are manually applied with the judgment and skill of the designer [1].

While still useful, there is a need for more structured DFT techniques that can be automatically applied to a design using today's design, synthesis and test tools. The only systematic DFT approach that can be easily incorporated in the overall design process, enhances the controllability and observability of a module, and requires few extra test pins, is known as *scan design*.

A scan design requires the compliance with a set of ground rules centered around uniform design methodologies for state variables (flip-flops or latches). It is quite clear that if all the state variables are directly controllable and observable, then testing scannable circuits is reduced to testing just combinational circuits.

To provide a sample of structured scan designs, the following sections describe three different widely used design styles: level-sensitive scan design, scan path, and boundary scan design. While they were originally developed within the framework of conventional test on external testers, they are equally applicable and important for built-in testing.

2.1.1 Level-Sensitive Scan Design

A Level-Sensitive Scan Design (LSSD) is probably one of the most used and best documented structured DFT technique [1, 3]. Figure 2.1 shows the separation of the combinational logic and the sequential elements in an LSSD structure.

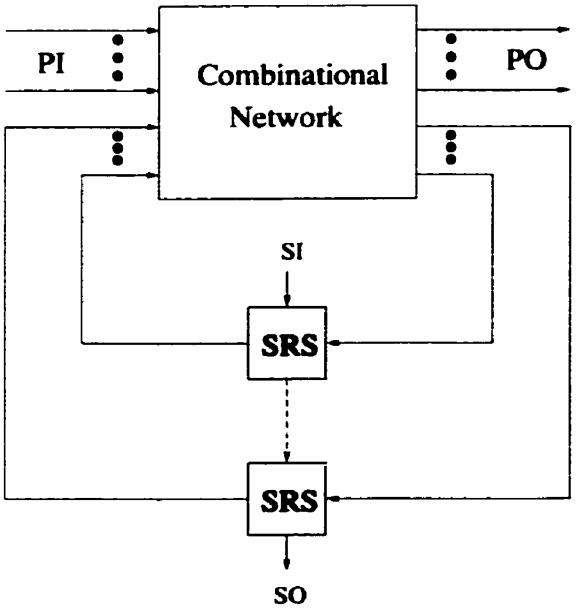


Figure 2.1: The LSSD sequential circuit model [1].

The Primary Inputs (PIs), Primary Outputs (POs), and combinational circuits are not modified. The state elements are implemented as clocked D-latches ($L1$). Each latch is augmented to form a Shift-Register Stage (SRS) by adding a second latch ($L2$) fed by $L1$. A multiplexer at the input of the $L1$ latches selects between the system inputs in normal mode, and the Scan-In (SI) inputs in test mode. The SI input and the $L2$ Scan-Out (SO) output are, respectively, the input and output of the SRSs. All SRSs are chained together during test into one or more shift-register chain(s). Interconnecting the SRSs into a shift-register chain is done by connecting the SO output of one SRS to the SI input of the next, and by connecting all shift clocks in parallel [1]. Accessing the SRSs in test mode is done through the chains SI primary inputs and the SO primary outputs. Each shift cycle moves the test data one step down the shift-register chain. The $L1$ and $L2$ shift clocks must be non-overlapping to ensure a correct shift operation.

A minimum of four additional package pins are needed to implement LSSD. Two pins are used for the SI and SO ports of the shift-register chains, and two are used for the $L1/L2$ shifting clocks. Additional pins are required if multiple scan chains are used.

During normal system operation, the $L1/L2$ shift clocks are off and the SRS has just two input signals, system data and system clock (CLK). When CLK is on, the internal state of $L1$ is changed to the value of the system input.

During test, the test patterns scanned into the chain(s) are applied to the combinational logic at the SRS outputs, and the response values captured in the SRSs are scanned out for examination. Thus, the state elements become *pseudo-inputs* for the purpose of test application and *pseudo-outputs* for test observation [1]. The process of applying test stimulus and observing its response is overlayed: as the stimulus for vector "t" is shifted into the scan path, the response for vector "$t-1$" is shifted out.

2.1.2 Scan-Path Design

The scan-path architecture is another widely used DFT technique, that increases the observability and controllability of the internal nodes in a synchronous circuit [1, 3].

The scan-path structure is very similar to LSSD in that it implements state variables as stages of a shift register for scanning test-data in and test-responses out. The scan path register stages, however, are built with D flip-flops. The operation of a scan path circuit requires two clocks: CLK used during normal system operation, and TCLK used during scan shift operations. Thus, only three additional pins are needed to implement a scan path design.

2.1.3 Generic Boundary Scan Architecture

Boundary Scan (BS) is another structured DFT technique applicable to digital modules [43, 44]. The BS strategy was advanced by the Joint Test Action Group (JTAG) [58], that began in Europe and spread quickly to the United States and Japan. The JTAG proposal for a boundary scan implementation was adopted formally as the IEEE standard 1149.1 in 1990 [59]. The standard has been endorsed by many leading North-American, Asian and European companies [2]. The BS technique can be used in the design of large, complex ICs, MCMs, loaded boards, and complex systems. As a result, the technique looks to be set to become one of the major techniques of testing complex digital modules [44].

To implement BS, a scan SRS is placed on every primary input and output of a module, that is, at its boundaries. To achieve this, a specialized test circuitry needs to be added to the module between each external I/O pin and the logic to which it is connected to, except for clocks and test control pins. This does not imply that all internal state elements in a module must be scan-testable.

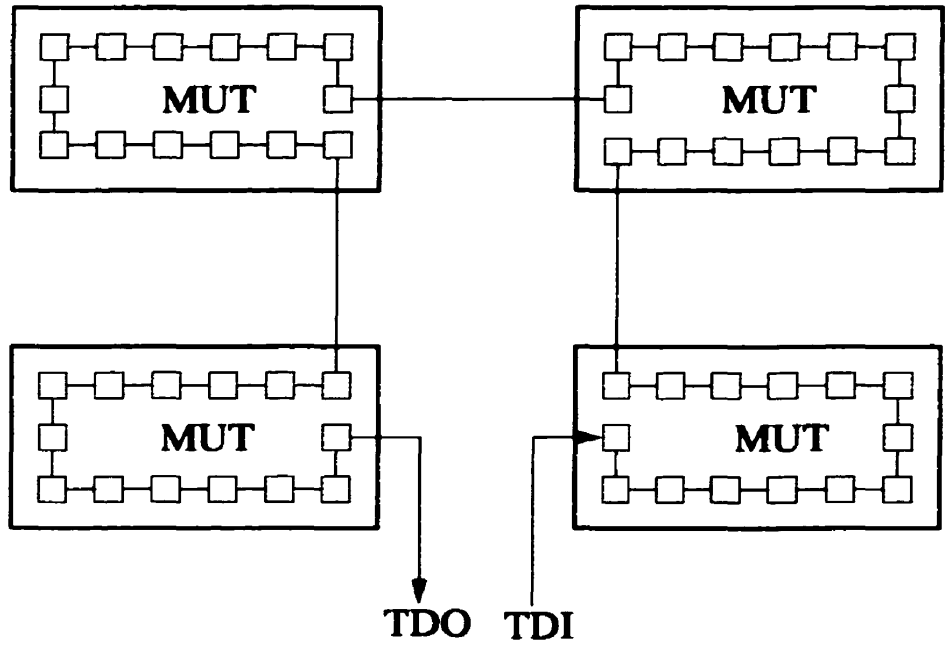


Figure 2.2: Board-level BS Scenario.

These test circuits, called Boundary-Scan Cells (BSCs), are connected into a shift register path around the periphery of the module. The boundary-scan path can be considered as a very wide parallel load, serial shift register.

If a module is constructed entirely from components built using boundary scan, the BS paths of all devices can be cascaded during test to form a single large boundary-scan path. A test system can then control and observe all device pins and their associated interconnects through this path [52].

Figure 2.2 illustrates how individual BS components can be connected at the board level and brought to the edge connectors. The scan output (TDO) from one chip is connected to the scan input (TDI) of the next chip in the scan chain.

A scan tester [22, 45] clocks data into and out of the BS path via the *serial* I/O test pins to perform various tests. The path provides virtual test channels on the BS device pins [60]. Thus a virtual bed-of-nails capability is achieved, easing the testing and diagnosis problems in VLSI modules.

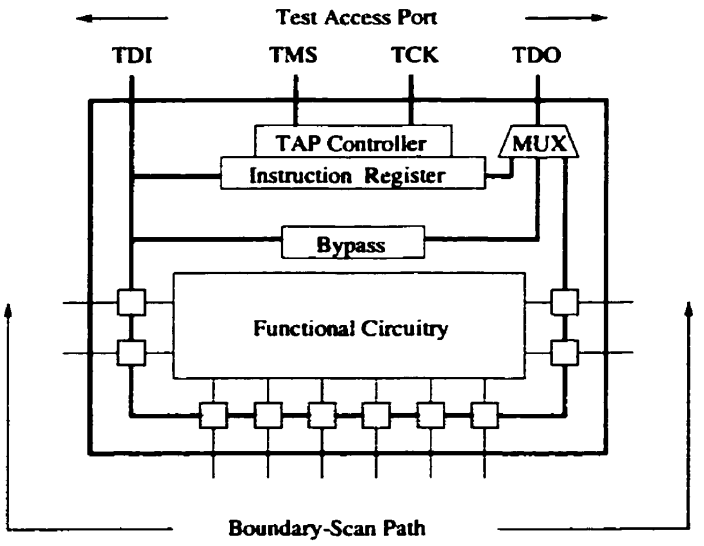


Figure 2.3: The IEEE boundary-scan architecture for one chip [2].

Boundary Scan at the Chip Level

The IEEE boundary-scan architecture for one chip is shown in Figure 2.3.

Four pins, constituting the *serial* Test Access Port (TAP), are added to each chip to make it scan testable. These pins include : Test Data Input (TDI) which allows the chip to receive test data, Test Data Output (TDO) which shifts out the chip's output data, and Test Mode Select (TMS) which along with the Test Clock (TCK) define the test mode to be executed. This mode is stored in the Instruction Register (IR). Depending on the content of the IR, the Boundary Scan Register (BSR) surrounding the functional circuitry of the chip, a User Defined Register (UDR), or the 1-bit Bypass Register (BR) is selected.

The 1-bit BR is used as a shortcut path through the chip in order to shorten the overall scan path length, thus saving time and test-equipment resources when testing other chips on the module. Each UDR allows the chip to operate in a user-defined test mode, such as BIST.

The TAP controller is a Finite State Machine (FSM) that controls the operations sequence in the BS circuitry. By means of the TMS and TCK pins, the BS components are moved through 16 TAP states.

During normal operation of the component, data flows directly through the BSCs to exercise the component's internal logic. In test mode, a scan operation is composed of

three states: CAPTURE, SHIFT, and UPDATE [44].

During a CAPTURE state, the data or instruction bits are loaded in parallel into their corresponding registers. The data bit values can come from either the component itself or the external logic connected to the BS cells. The new data or instruction takes effect and becomes the current data or instruction at the UPDATE state. Finally, during the SHIFT state, the existing data in the selected register is shifted out through the TDO pin and a new data is shifted in from the TDI pin.

There are three test modes for the boundary-scan cells [43, 44, 59]. In the *internal* chip test mode, the BSCs are used to simulate the chip's internal logic with test-data and to capture its test-responses.

During *external* testing, the BSCs at the output pins are used to drive test values into the external connections, while those at the input pins capture the received values and shift them out for examination. By a careful selection of test patterns, the interconnections between BS-compatible ICs can be easily tested for stuck-at, shorts, opens, and other fault types.

Finally, a *sample* test is also possible, allowing engineers to take a snap-shot of the signal values received-by or sent-from the chip during the normal operational mode. This test represents an excellent feature that can be used for improving debug and diagnosis.

Separating the tasks of component and loaded module testing offers a significant advantage. It allows the reuse of most existing test data generated for the component level at the module level. A test applied to a module can be constructed from tests generated for the single components [47]. This is a notable contrast to the case without boundary-scan, where the test engineer needs to know a great deal about the component's operation in order to generate either an in-circuit or a functional module test [43]. These hierarchical properties will become critical as MCMs and wafer scale integration continue to push entire boards into single packages [61].

2.2 Built-In Self-Test

With the advent of denser technologies, the balance is shifting decidedly in favor of testable designs [62]. The ultimate in a testable design is to make the design test itself, and hence the notion of Built-In Test (BIT).

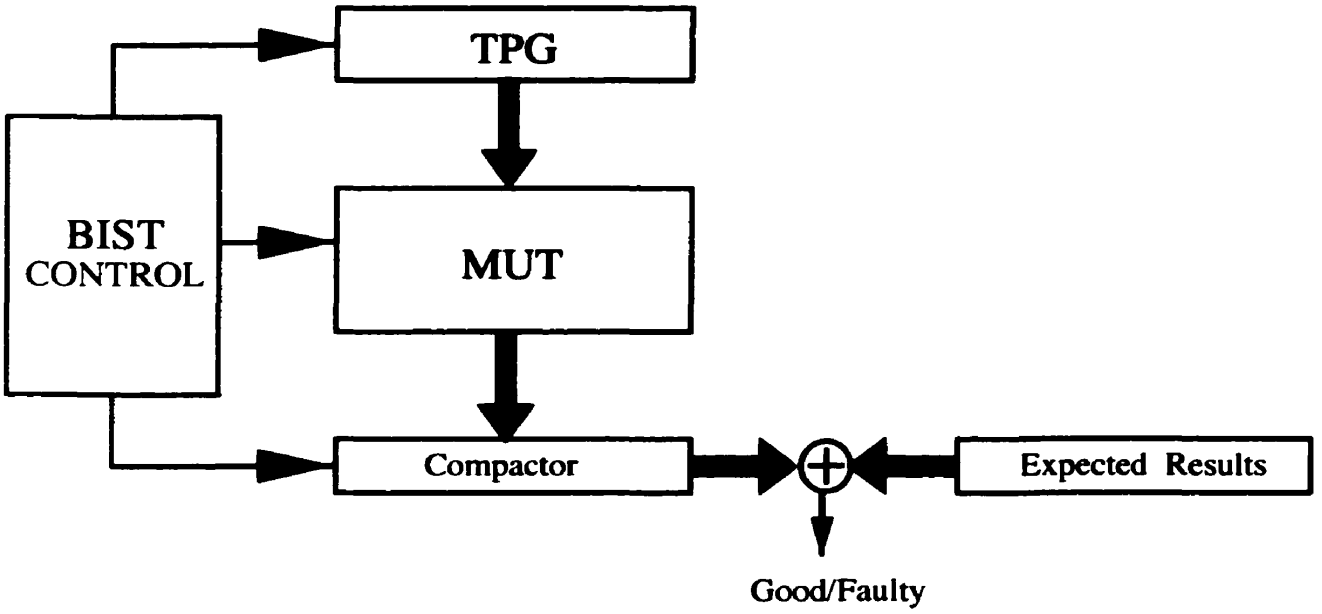


Figure 2.4: General test setup in a BIST environment.

BIT can be either concurrent or non-concurrent. Concurrent or implicit testing, sometimes also called checking, refers to the on-line testing using either hardware redundancy, information redundancy or both, to detect errors that occur during normal module operation [1]. It includes methods such as error detection and correction circuits, totally self-checking circuits, and self-verification. Concurrent testing is beyond the scope of this thesis. The discussion will be restricted to only non-concurrent testing and the terms testing and test will be used to refer to non-concurrent testing.

Non-concurrent or explicit testing is carried out while the MUT is not in use. It can be either functional or structural in nature. Non-concurrent BIT is often referred to as BIST, a term that will be used throughout the thesis.

For today's complex VLSI modules, BIST has been shown to be more effective than the traditional external test methodologies [38]. BIST has been warmly embraced by the IC industry [14, 63, 64] as one of the most promising solutions for testing tomorrow's ULSI modules in deep sub-micron [15].

Various approaches to BIST have been successfully explored in the literature [1]. Figure 2.4 shows the general structure of BIST.

In BIST, both Test Pattern Generation (TPG) and output response analysis are conducted on the same module as the MUT. To analyze the test responses efficiently in silicon,

BIST schemes usually employ a data compactor to compress the large amount of responses into a small *signature* of a few bits [1].

The TPG in Figure 2.4 supplies the test patterns to the MUT. A "*start*" signal is issued by the BIST controller with the generation of the first test pattern. This signal tells the data compactor to start compressing the output responses. The compaction goes on until test completion time, where a "*stop*" signal is issued by the BIST controller after applying the last test pattern. A stop signal to the compactor constitutes a "*ready*" signal for the comparator. The compacted signature is compared to its "*golden*" reference, derived from a good-machine simulation. The comparison may be done off-line by unloading the signature register for comparison outside the MUT, or on-line by comparing the signature to a hard-wired golden one within the MUT itself. The MUT passes the test if the signatures are identical.

2.2.1 BIST General Goals

Since the advent of BIST techniques, several goals have been developed and used to benchmark and compare the different BIST approaches [65]. Some of the most important ones are listed below.

The first and foremost goal is that of providing high fault coverage as would be expected of any testing approach [65]. Some BIST approaches guarantee high fault coverage through exhaustive or pseudo-exhaustive test pattern generation [66]. Other approaches cannot guarantee high fault coverage but give acceptably high fault coverage in practice [67].

Obviously, BIST techniques are not free. They require an investment in extra module area for the realization of the test circuitry. The second BIST goal is that of a low area overhead that minimizes the BIST implementation cost. Typical area penalties range from 15% to 40% [68]. Even with these penalties, it is interesting to note that BIST is still predicted to be the cheapest testing option if the total lifetime test costs are considered [62].

A third goal is the desire for little, if any, performance penalty to the module function as a result of BIST incorporation. Typically, BIST adds two to three gate delays to the paths in which the test circuitry is implemented [65]. Performance is degraded if the BIST circuitry is included in the critical timing path of the module.

An additional goal is the ease with which a particular BIST technique can be imple-

mented and understood. BIST techniques that lend themselves to design automation are more attractive.

The final goal is that of providing *vertical* testability such that a BIST capability can be used at all levels from component level testing through system level testing. To accomplish this, system BIST must be able to communicate with subsystem level BIST, which in turn must be able to communicate with component level BIST. When this communication is possible, BIST at one level can invoke a lower (deeper) level BIST, thus achieving the ability to perform a *hierarchical* fault isolation [61, 69].

2.2.2 BIST Advantages

Despite the fact that BIST consumes extra area and has an I/O overhead, it also results in visible reductions in the testing costs when compared to an external test using automatic test equipment [1]. BIST achieves these savings in a variety of ways.

First, it reduces the cost of test pattern generation. BIST does not require that patterns be stored in the test equipment. Thus, testers can be less expensive due to the reduction in the needed memory [47]. This feature is especially important in a high-performance module testing [63].

BIST, by definition, executes at the normal speed of the MUT [9]. Running tests at-speed shortens the test application time, when compared to the usually slower, external tester approach. In addition, it also provides a convenient way of applying more test patterns to compensate for the weakness of the abstracted fault models [70].

The on-module TPG and ORA alleviate, or even eliminate the need for expensive, sophisticated external testers at all levels of packaging [9]. The tester simply provides a clock and a few control signals, and it contains a limited memory space. The on-module test capabilities also make a module more independent of the specific test resources available at each packaging level [63]. This in-turn makes adapting to technology changes easier.

Linking BIST to a diagnostic software reduces the software complexity and increases the diagnostic accuracy [30]. This feature saves a lot of the man and computer power that would have been needed to develop complex diagnostic software tools, and drastically reduces the system run time for fault isolation [47].

Finally, a hierarchical BIST technique provides better fault detection and faster fault isolation, while amortizing the BIST cost over a wide range of applications. It also offers

system benefits such as reduced diagnostic code development time, diagnostic run time, maintenance time, and system down time [71, 72].

There are three main factors which determine the effectiveness of any BIST scheme: the input test patterns applied to the MUT, the nature of the MUT and its fault mechanisms, and the choice of the data compaction technique [73]. These factors will be explored subsequently in Sections 2.3 and 2.4.

2.3 BIST Test Pattern Generation

In a BIST environment, there are many ways of generating the test patterns for the MUT. The simplest categorization, in terms of the type of testing used, is: pre-stored testing, exhaustive testing, and random testing [1].

In pre-stored testing, the test patterns are generated using Automatic Test Pattern Generation (ATPG) tools, and are stored in a ROM [73]. This procedure may not be able to cope with the combinatorial growth associated with large modules.

In exhaustive testing, all possible test patterns are applied to the MUT. Thus, the exhaustive test length is 2^m tests, where m is the number of inputs to the module. The tests can be generated using an m-stage binary or gray counter, or a *nonlinear* (de Bruijn) feedback shift register [1]. The exhaustive testing of a MUT, with a large number of inputs, requires large TPG registers and relatively long test application times.

Another TPG approach that has been the most effective for structural BIST testing, is that of random pattern generation. This will be discussed next.

2.3.1 Pseudorandom Testing

Random pattern testing has been a common practice in the IC industry for a long time [1]. It entails the application of a randomly chosen subset of all possible input patterns. Since an ideal random number generator does not exist, then a truly random test cannot be generated. However, it is relatively easy to generate a *Pseudo-Random* (PN) test as an approximation to the truly random one.

A binary sequence of 1's and 0's is called pseudorandom when its bits appear to be random in the local sense, but they are fully repeatable, and hence only pseudorandom.

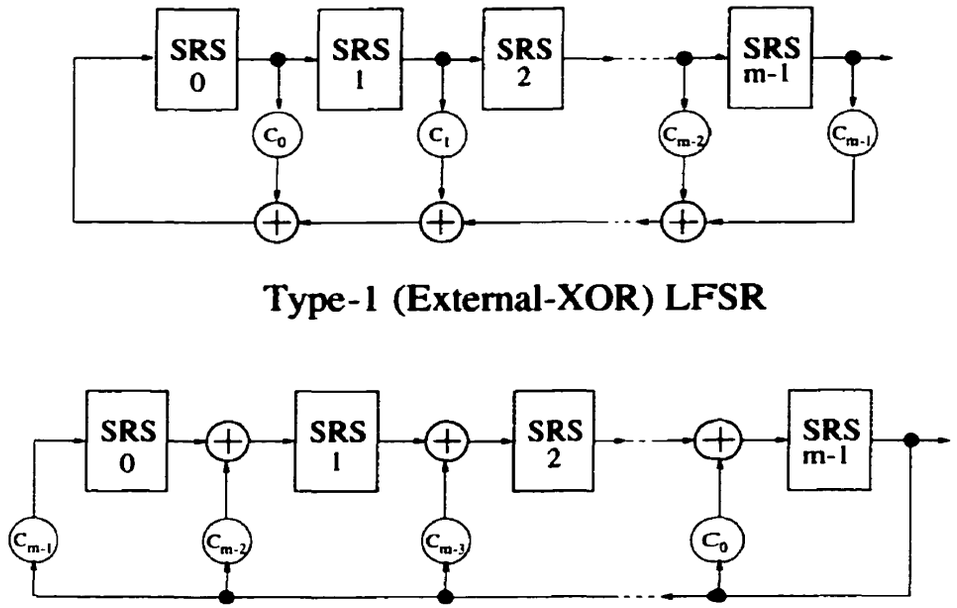


Figure 2.5: Canonical LFSR structures [3].

These sequences have been studied extensively in [74, 75]. For test applications, this is an advantage, since repeatable tests ensure repeatable signatures [1].

2.3.2 Shift-Register Implementation of a PN Sequence

Autonomous feedback shift registers have been widely used to generate pseudorandom sequences [1]. A shift register with a *linear* feedback network is called a Linear Feedback Shift Register (LFSR). A linear binary network is constructed from only unit delays (D-type storage cells), modulo-2 adders (XOR gates), and modulo-2 scalar multipliers (1 or 0).

An LFSR, which has the canonical forms shown in Figure 2.5, is the usual choice for a pseudorandom test generator. Its sequential output words, while deterministically generated, appear random in the local sense and will pass most common tests for randomness [1].

Let the sequence $\{a_i\}_{i=0}^{\infty}$ represent the sequence generated at the output stage of an LFSR, where $a_i \in \{0, 1\}$ is the state of the output stage at time t_i. The m^{th}- degree

polynomial,

$$f(x) = 1 - \sum_{i=0}^{m-1} c_i x^{i+1} \ , \tag{2.1}$$

is referred to as the *characteristic polynomial* of the sequence $\{a_i\}$ and the LFSR that produces it. Since subtraction is the same as addition in modulo-2 arithmetic, then,

$$f(x) = 1 + \sum_{i=0}^{m-1} c_i x^{i+1} \ , \tag{2.2}$$

where $c_i = 0, 1$ is implemented as either a connection or no-connection, respectively. Thus, for every characteristic polynomial, two LFSR implementations are possible.

Since an LFSR is a finite state machine, then each state is uniquely determined from the previous state by the feedback connections. Given an m-stage shift register, there are at most 2^m possible states. Since the feedback network is linear, then the successor of the all-zeros state is itself. Hence, an LFSR can have at most $2^m - 1$ states when initialized with a non-zero starting seed. Consequently, the sequences generated at any of the LFSR stages are periodic with a period no greater than $2^m - 1$ [1].

The actual length of the sequence depends on both the number of register stages and on the details of the feedback connection. If the generated sequence has a period of $2^m - 1$, then it is called a *maximum length* sequence. The characteristic polynomial of a maximum-length sequence is called a *primitive polynomial* [1]. An extensive listing of primitive polynomials can be found in [76].

An LFSR can be used directly to feed a single input MUT. If the number of inputs in the MUT is $m \geq 1$, then m LFSRs can be used to feed the MUT. This might be very expensive to implement. An alternative approach is to use a single m-stage LFSR, and use the output at each stage to feed the MUT inputs directly. The only problem with this approach is that the generated sequences at adjacent stages are correlated, thus degrading the randomness of the generated tests [77]. This problem can be avoided by combining the LFSR with a linear XOR network (phase shifter [1]) to provide shifted, less correlated versions of the generated sequences. Alternatively, a carefully designed Type-2 LFSR with an internal feedback can be used [3].

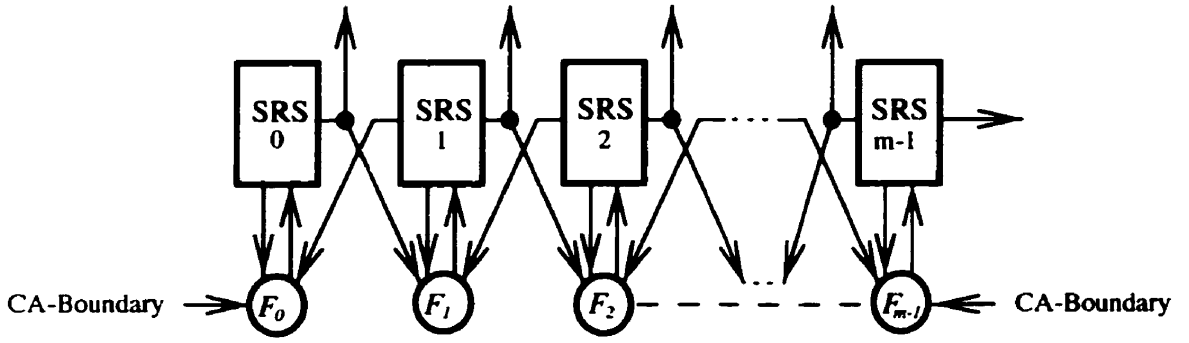


Figure 2.6: A 1-dimensional linear cellular automata.

2.3.3 Cellular-Automata Implementation of a PN Sequence

Cellular Automata (CA) is another FSM structure that exhibits pseudorandom generation traits. It consists of a number of cells arranged *spatially* in a regular fashion [78], where the state transition of a cell depends on the states of its neighbours. Each cell is made of a storage element (D latch or flip-flop) and some combinational logic implementing its next state function.

Because of its simple structure and local interconnections, CA is highly suitable for VLSI implementations. The most commonly used CA form in BIST environments is the 1-dimensional linear CA, with only local-neighbour interconnections. This class of CA has been extensively studied as a source of pseudorandom test patterns [79, 80]. For such CA, the next state of a particular cell depends only on itself and its two immediate neighbours. The next-state function employs only linear XOR/XNOR logic gates. If the two end cells of the automata are connected to a constant logic 0, it is said to have a *null boundary*. Figure 2.6 illustrates an m-cell 1-dimensional linear CA that can be used to feed an m-input MUT.

The state of a cell at time $(t+1)$ is given by:

$$Q_i(t+1) = \mathcal{F}\left(Q_{i-1}(t), Q_i(t), Q_{i+1}(t)\right) \quad , \tag{2.3}$$

where $Q_i(t)$ denotes the state of the i^{th} cell at time t, and $\mathcal{F}$ is the next state function called the *rule of the automata*.

If $\mathcal{F}$ is expressed in the form of a truth table, then the decimal equivalent of the binary outputs in the truth table is conventionally called the *rule number* of the CA. There are 256 possible rules. Rule 90 and rule 150 are the commonly used ones. Rule 90 means

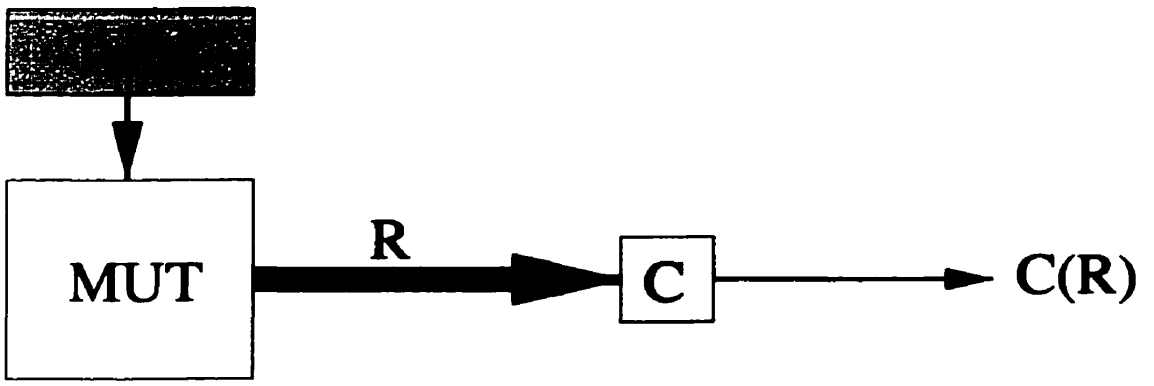


Figure 2.7: Generalized BIST compaction model.

$Q_i(t+1) = Q_{i-1}(t) \oplus Q_{i+1}(t)$, and rule 150 means $Q_i(t+1) = Q_{i-1}(t) \oplus Q_i(t) \oplus Q_{i+1}(t)$. A hybrid CA is constructed using more than one rule.

A linear CA is isomorphic to a conventional LFSR, and can also be represented by a characteristic polynomial [81]. Thus, a primitive m-cell CA cycles through $2^m - 1$ states when initialized with a non-zero starting seed. However, compared to an LFSR, CA serves as a better quality source of pseudorandom parallel test vectors [82]. This is due to the reduced correlations or linear dependencies between the bit streams generated at adjacent cells, thus resulting in superior randomness properties.

2.4 General Aspects of BIST Compaction Techniques

A high quality test is generally attained in BIST by applying a large number of test patterns to the MUT [1]. The difficulty in testing such modules resides in the excessive volume of test response data that must be stored. It is obviously unsatisfactory to build into the module a bit-by-bit comparison of its test responses with the expected reference ones, because of the large silicon overhead needed for storing the reference data. This makes output response compaction mandatory.

The usual Compaction method (C) is to capture and then compare some statistic $C(R)$, called *signature*, of the module output Responses (R) rather than comparing the individual responses themselves. $C(R)$ is usually chosen such that its value depends on all the response data, and the hardware required to implement C is simple [73]. The concept is illustrated in Figure 2.7.

Any data compaction method tends to lose information, and hence the term compaction and not compression. This loss is usually called *masking* and is measured by the probability of a faulty module producing the same signature as the fault-free one.

Several compaction techniques have been suggested in the literature, some of which are in actual use [1]. A unified treatment of some of these techniques is given in [73]. The choice of a certain method depends on the tradeoffs between its cost and its fault coverage or masking properties. All of these techniques fall in two main categories: *Time Compaction* (TC) and *Space Compaction* (SC).

Time compaction reduces *all* of the MUT's test responses into a fixed-width signature. Changing the test length, and thus the test application time, changes only the value of the final signature but not its width.

Space compaction on the other hand, reduces the *width* or *space* of the test responses for each test. It is usually combined with time compaction to reduce the number of MUT outputs that have to be compressed.

TC techniques include parity checking, transition counting, syndrome generation (or ones counting), Walsh spectra coefficients, output modification, and signature analysis [1, 3].

In parity checking, the parity of the test response sequence is checked using a simple XOR gate (or a tree of XOR gates) and a SRS. Transition counting, uses a counter and the parity checking circuitry to count the number of transitions in the output response. The syndrome of a module is the number of times a logic "1" is produced at its output. Thus, the syndrome generator is a simple counter. Closely related to syndrome generation are the Walsh spectral coefficients, and output modification. In both cases, a sequence is XORed with the test response sequence and the syndrome or weight of the sum is determined. Finally, in signature analysis, LFSR structures have been widely used in BIST environments to compact the test responses [1].

A detailed analysis of the above techniques [1], clearly favors signature analysis as a practical, high-quality BIST compaction technique. Henceforth, only signature analysis will be considered in this thesis. A summary of the method will be given first in Section 2.5. The additional use of space compaction will be discussed subsequently in Section 2.6.

2.5 Signature Analysis In BIST

The TC scheme that has thus far received the most attention is known as *signature analysis*, a name coined by Hewlett-Packard [83].

The operational principle of a signature analyzer is based on a restricted class of data-transmission parity-codes, called Cyclic Redundancy Codes (CRC) [84]. CRC coding has been implemented successfully in SA using linear shift registers with special feedback structures.

The test responses from the MUT are fed into the shift register in synchronization with the shifting clock. The *residue* or *checksum* left in the register, after feeding the last test response, is the signature.

SA is generally the best single compaction method, because of its sensitivity to the 1's and 0's in the TRS, and because it can be easily modified for use with multiple-output modules [1].

2.5.1 LFSRs as Signature Analyzers

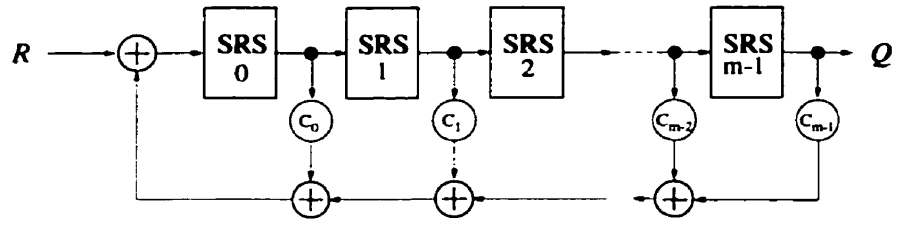
LFSRs have been widely used in BIST and in external testers as signature analyzers [1]. A single-input LFSR can be used to compact the responses from a single-output MUT.

Again, two general structures can be considered for implementing the single input signature analyzer: the External-XOR (Type-1 LFSR), and the Internal-XOR (Type-2 LFSR). Figure 2.8 shows the general structures for an m-stage LFSR compactor.

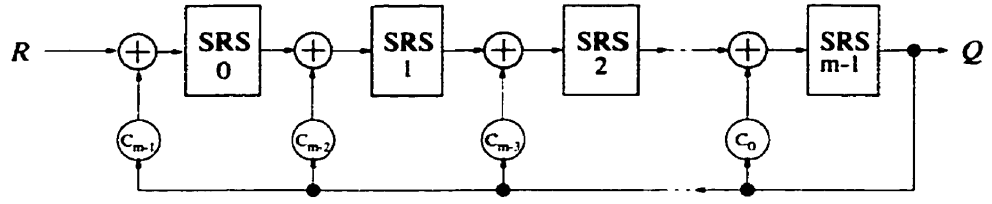
The Type-2 structure may be inconvenient for implementation because of the need for XOR gates between the shift register stages. The choice between the two implementations depends upon the available circuits and their fault detection/diagnosis properties, as will be explained throughout the thesis.

Before starting a test, the LFSR is initialized to a certain seed to ensure repeatable signatures. Test responses are applied to the LFSR synchronously with the shifting clock. After the responses from the last test are shifted-in, the final content of the LFSR is the resultant TC signature.

Thus, the LFSR generates the *remainder* "code word" after dividing the test response sequence. The divisor polynomial for an LFSR is determined by its feedback connections,



Type-1 (External-XOR) LFSR



Type-2 (Internal-XOR) LFSR

Figure 2.8: Canonical single-input signature analyzers.

which in turn determine its properties. The feedback connections are specified by the characteristic polynomial:

$$f(x) = 1 + \sum_{i=0}^{m-1} c_i x^{i+1} \ , \tag{2.4}$$

where x^i is a delay operator and c_i is a constant binary multiplier (1 or 0) implemented as either a connection or no connection to the feedback path, respectively. The derivation of this polynomial can be found in [74].

The LFSR divisor polynomial ($G(x)$) is the *reciprocal* of its characteristic polynomial, where the reciprocal polynomial ($f^{-1}(x)$ or $f^*(x)$) is defined as:

$$f^{-1}(x) = x^m f(\frac{1}{x}) \ . \tag{2.5}$$

The signature *space* of an LFSR corresponds to the vector space of its characteristic polynomial. The maximum signature space is obtained with a primitive polynomial as explained earlier in Section 2.3.2 . It is a property of characteristic polynomials that the reciprocal of a primitive polynomial is also a primitive polynomial [1].

The choice of the feedback polynomial affects the LFSR's error detection/location capabilities. A suitable polynomial must be chosen from various classes of error-correcting/error-detecting codes to cover the expected error distributions [73].

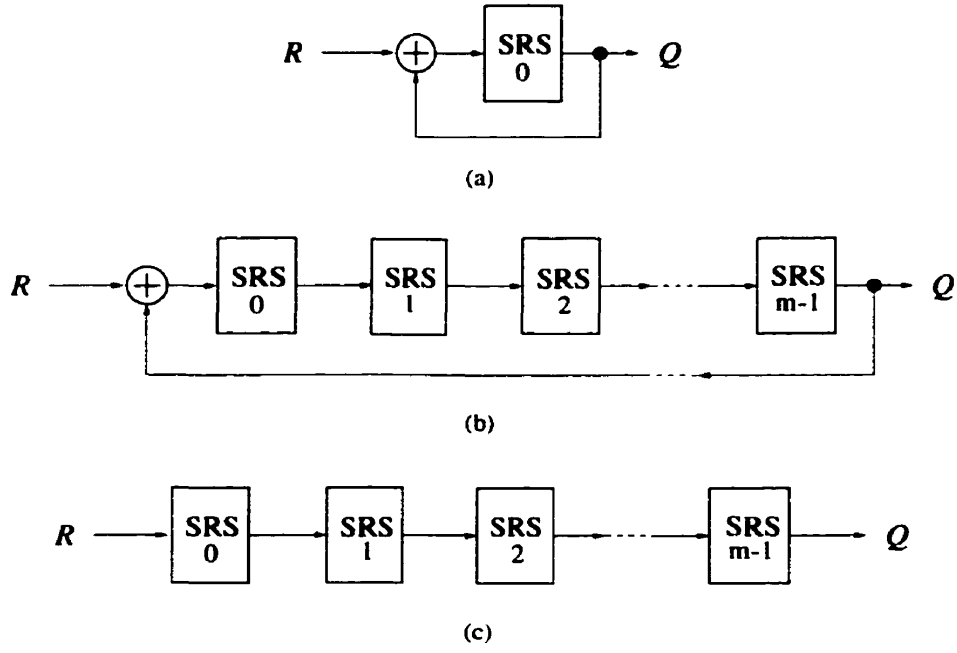


Figure 2.9: Special linear shift-register structures.

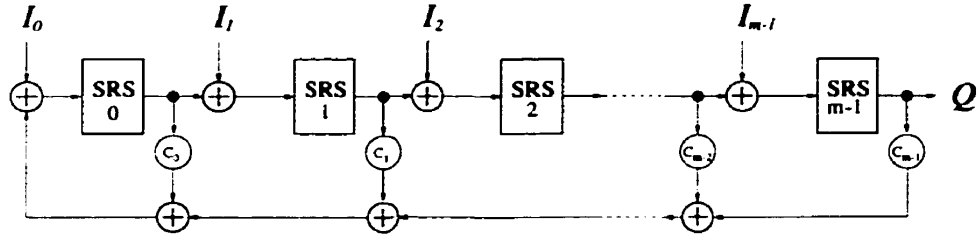
2.5.2 Special Feedback Structures

Consider the three special linear shift-register structures shown in Figure 2.9.

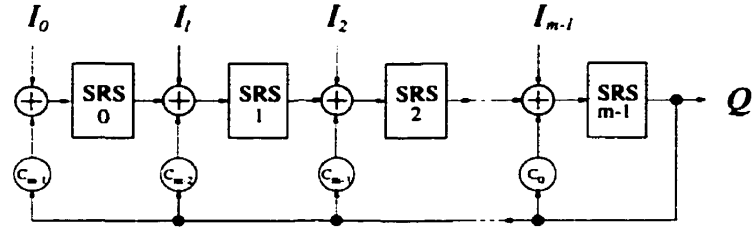
The first LFSR in Figure 2.9(a) implements $f(x) = 1 + x$. This corresponds to the case when a single output of the MUT is compacted to a single bit signature. This signature is simply the parity of the TRS, just like in parity compaction.

The register of Figure 2.9(b) is a simple LFSR whose only feedback is from the last stage to the first stage. It is a perfectly adequate signature analyzer that implements division by $1 + x^m$. Despite its simplicity, it is just as good as the one having a more complicated feedback [85]. It is usually called a pure *cycling* register because in the absence of an input, the register merely cycles its contents. As a result, each bit in the cycling register is independent of the others [85]. When used as a signature analyzer, each bit in the signature is just the parity over every m^{th} bit of the TRS [1].

Finally, it is interesting to consider the ultimate case of the register shown in Figure 2.9(c) having no feedback at all. The divisor polynomial is x^m. The signature in this Linear No-Feedback Shift Register (LNFSR) is simply the last m TRS bits shifted in.



Type-1 (External-XOR) MISR



Type-2 (Internal-XOR) MISR

Figure 2.10: Generic multiple-input signature analyzers.

2.5.3 Multiple-Input Signature Analyzers

For BIST of multiple-output modules, the overhead of a single-input signature analyzer on every output would be high. Of course, the single-input analyzer can be time-multiplexed to the various MUT outputs [1]. This will require repeating the test sequence for each output, resulting in an excessively long test application time.

Using a *parallel* signature analyzer can significantly reduce the test time. The parallel testing method using a structure called Multiple-Input Signature Register (MISR) is the preferred BIST compaction technique for multiple-output modules [86, 87, 88]. The general structures for an m-bit MISR are shown in Figure 2.10.

Usually, a MISR has as many stages as there are module outputs. The m available MISR inputs are connected to the m MUT outputs. If the MUT has fewer outputs than m, the remaining free MISR inputs are connected to a constant logic 0.

Similar to an LFSR, a MISR is also characterized by the characteristic polynomial $f(x)$ in Equation 2.4 . However, it should be noted that $f(x)$, which up to this point has been represented as an m-degree polynomial over the binary Galois Field (GF(2)), has an equivalent representation [89] over GF(2^m) as the 1-degree polynomial:

$$f(x) = x + \alpha \ , \tag{2.6}$$

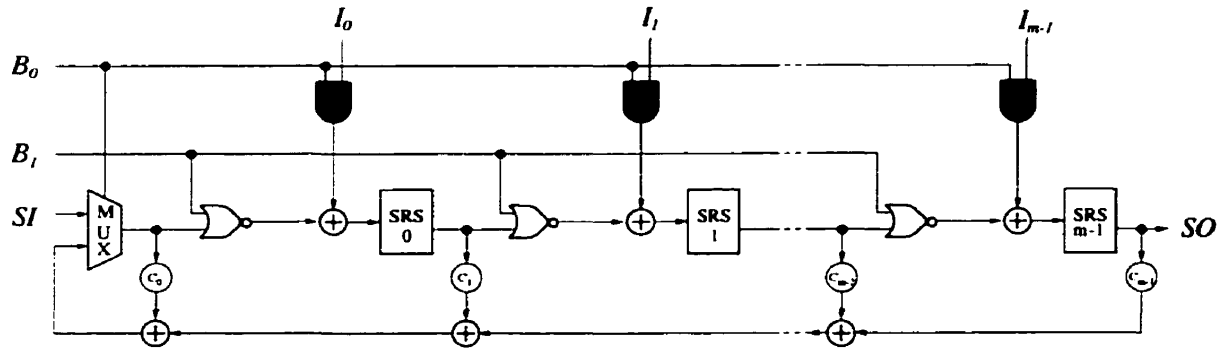


Figure 2.11: The BILBO register.

where $\alpha \in \mathrm{GF}(2^m)$. The m-bit MUT outputs can thus be interpreted as elements over $\mathrm{GF}(2^m)$ in this representation.

Finally note that the special LFSR structures presented in Section 2.5.2 have equivalent MISR implementations.

2.5.4 Built-In Logic-Block Observation

The Built-In Logic-Block Observation (BILBO) is a special BIST structure that can simultaneously generate pseudorandom test patterns and compact test responses [1].

The BILBO register shown in Figure 2.11 has four operating modes that can be selected using the B_0 and B_1 inputs. If $B_0B_1 = 00$, the BILBO becomes a shift register with scan input SI and scan output SO. The register is cleared if $B_0B_1 = 01$. When $B_0B_1 = 10$, it becomes either a MISR or an LFSR. If the I_i's are connected to the MUT, then the BILBO acts as a MISR. The compacted signature can then be shifted out for examination by selecting the shift mode. However, if the BILBO inputs are held constant, then it acts as an LFSR generating pseudorandom test patterns. Finally, when $B_0B_1 = 11$, the BILBO behaves as a parallel-input, parallel-output register. This is the normal mode of operation. Using this mode together with the shift register mode enables scan testing.

Figure 2.12 illustrates a module with BILBO registers. BIST testing can be performed in two steps, by configuring the BILBO registers to the left and right of CUT_i as an LFSR and a MISR, respectively. The test sequence is repeated for each CUT_i.

Note that the testing can be done in one step if BILBO_1 is configured as an LFSR, and BILBO_2, BILBO_3 are configured as MISRs. In this case the compacted responses of CUT_1

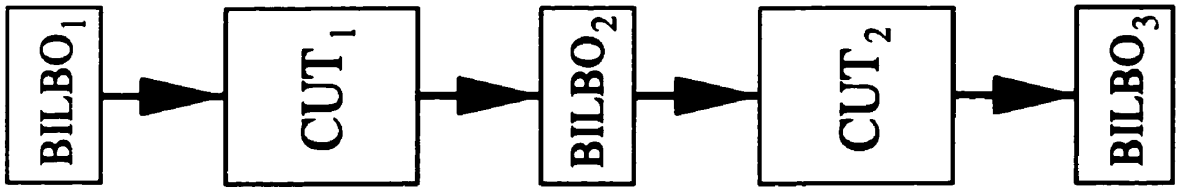


Figure 2.12: BIST testing using the BILBO architecture.

are used as test stimuli to CUT_2.

2.5.5 Polynomial Representation of SA

Consider an m-output MUT, where $m = 1$ for a single output MUT. In test mode, the Signature Analysis Register (SAR) is first initialized, and the output responses of the MUT are compacted serially into a signature. Let the output responses be represented as:

$$R = \{R_0, R_1, \ldots, R_{T-2}, R_{T-1}\} , \tag{2.7}$$

where T is the number of tests applied to the MUT, and $R_t \in \mathrm{GF}(2^m)$ $(0 \leq t < T)$ is the response for the t^{th} test pattern.

Coding theory [84] treats streams as polynomials in a dummy variable. Using the dummy variable x, the MUT outputs can be converted into a polynomial in x by letting each R_t to be the coefficient of a unique power of x [1]. Thus, the corresponding TRS polynomial can be written as:

$$R(x) = R_0 x^{T-1} + R_1 x^{T-2} + \cdots + R_{T-2} x + R_{T-1} . \tag{2.8}$$

Note that R_0 is assigned the highest power in the polynomial because it is entered first in the SAR.

This transformation permits the mathematical manipulation of TRS in polynomial form. If the SAR is initialized to zero and the test response words (R_t) are serially streamed to the analyzer input(s), then the content of the SAR after feeding the last test response R_{T-1} is the remainder from dividing the TRS polynomial $R(x)$ by the divisor polynomial $G(x)$ of the SAR [1]. The SAR emits the Quotient polynomial ($Q(x)$) by shifting it out from the right-hand end of the register. The remainder in the SAR is the Signature ($S(x)$) of the MUT. In mathematical terms,

$$\frac{R(x)}{G(x)} = Q(x) + \frac{S(x)}{G(x)} , \tag{2.9}$$

where the degree of $S(x)$ is less than the degree of $G(x)$. Equivalently,

$$R(x) = Q(x)G(x) + S(x) \ . \tag{2.10}$$

It should be noted that a Type-1 signature analyzer generates the correct quotient but the remainder is not always the correct one that would be expected from a true polynomial division [1]. However, it can be shown using the matrix formulation of Section 2.5.6, that the signatures obtained with a Type-1 and Type-2 analyzers are mathematically related. Therefore knowing one of them, the other one can be obtained easily with a matrix transformation.

In general, if the SAR is not initialized to the all zero state, then the m-bit signature is given by:

$$S(x) = [R(x) \bmod G(x)] \oplus k(x) \ , \tag{2.11}$$

where "mod" is the modulo-division operator without carry and $k(x)$ is a function of the initial state of the analyzer. If the initial state of the SAR is zero, then $k(x)$ is also equal to zero, and

$$S(x) = R(x) \bmod G(x) \ . \tag{2.12}$$

In the remainder of the thesis, it will be assumed without loss of generality that the SAR is put in the all zero state before the compaction process is started. If that is not the case, then the initial state can be thought of as being the test response R_{-1} corresponding to a fictitious test applied at time $t = -1$ to a cleared SAR.

Now, let the output response of a fault-free module be represented by the sequence:

$$R^0 = \{R_0^0, R_1^0, \ldots, R_{T-2}^0, R_{T-1}^0\} \ . \tag{2.13}$$

The polynomial associated with this response is given by:

$$R^0(x) = R_0^0 x^{T-1} + R_1^0 x^{T-2} + \cdots + R_{T-2}^0 x + R_{T-1}^0 \ , \tag{2.14}$$

and the fault-free signature $S^0(x)$ is equal to:

$$S^0(x) = R^0(x) \bmod G(x) \ . \tag{2.15}$$

If the MUT is faulty, then its output response will produce a signature that differs from the error-free one (neglecting masking). Let E be the error sequence corresponding to the observed output response. Then,

$$\begin{aligned} E &= R^0 \oplus R && (2.16)\\ &= \{R_0^0 \oplus R_0, R_1^0 \oplus R_1, \ldots, R_{T-2}^0 \oplus R_{T-2}, R_{T-1}^0 \oplus R_{T-1}\} && (2.17)\\ &= \{E_0, E_1, \ldots, E_{T-2}, E_{T-1}\} \ , && (2.18) \end{aligned}$$

and,

$$E(x) = E_0 x^{T-1} + E_1 x^{T-2} + \cdots + E_{T-2} x + E_{T-1} \ . \quad (2.19)$$

The signature $S_e(x)$ associated with error sequence is thus equal to:

$$\begin{aligned} S_e(x) &= E(x) \bmod G(x) && (2.20)\\ &= [R^0(x) \oplus R(x)] \bmod G(x) && (2.21)\\ &= [R^0(x) \bmod G(x)] \oplus [R(x) \bmod G(x)] && (2.22)\\ &= S^0(x) \oplus S(x) \ . && (2.23) \end{aligned}$$

The above relation was obtained by making use of the linearity of the XOR operation in SA.

For each unique error polynomial of degree d, less than the order of the SAR's divisor polynomial, a unique error signature will be produced [1, 90]. The order of $G(x)$, denoted by $\mathrm{ord}(G(x))$, is the least positive integer i for which $G(x)$ divides $x^i - 1$. If $G(x) = x^j H(x)$, then $\mathrm{ord}(G(x)) = \mathrm{ord}(H(x))$ [91].

Therefore, the number of possible signatures is equal to the order of the SAR's divisor polynomial. As the degree of the error polynomial exceeds the order of the divisor polynomial, the signatures begin to repeat in the same order as if the error polynomial was of a degree $d \bmod \mathrm{ord}(G(x))$ [90].

A MISR as a Single-Input Signature Analyzer

Consider again an m-output MUT whose test responses are being compacted by an m-stage MISR. The first attempts to study MISRs were based on replacing the m-input sequences by an equivalent 1-input sequence applied to the first input of the MISR [92]. This reduces the MISR structures to equivalent LFSRs.

The output response $R_t \in \mathrm{GF}(2^m)$ corresponding to test t can be equivalently represented in GF(2) as:

$$R_t = r_{t,0} r_{t,1} \cdots r_{t,m-2} r_{t,m-1} \ , \tag{2.24}$$

where $r_{t,i} \in \mathrm{GF}(2)$.

Let I_j $(0 \leq j < m)$ be the bit sequence entering the j^{th} input of the MISR. Then,

$$I_j = \{r_{0,j}, r_{1,j}, \ldots, r_{T-2,j}, r_{T-1,j}\} \ , \tag{2.25}$$

and,

$$I_j(x) = r_{0,j} x^{T-1} + r_{1,j} x^{T-2} + \cdots + r_{T-2,j} x + r_{T-1,j} \ . \tag{2.26}$$

It has been proven in [1] that if the MISR was initially cleared and the MISR is a true (Type-2) polynomial divisor, then the resulting signature can be expressed as:

$$S(x) = I(x) \bmod G(x) \ , \tag{2.27}$$

where,

$$I(x) = x^{m-1} I_{m-1}(x) + x^{m-2} I_{m-2}(x) + \cdots + x I_1(x) + I_0(x) \tag{2.28}$$

represents the modulo-2 summation of time-shifted copies of the individual $I_j(x)$.

Hence, the MISR signature is exactly the same as the one obtained from an LFSR having the same feedback connections as the MISR and $I(x)$ as its input.

The conventions developed in this section will be followed in the remainder of the thesis.

2.5.6 Matrix Representation of SA

The formulation derived next is based on the matrix representation of a signature analyzer. It was first used in [93] to formulate the signature generation process. It will be used below to study the faulty signatures of a MUT. The motivation is two-fold. First, it is believed that the matrix formulation serves as a more effective analytical tool in studying signature analyzers [64]. Analytical formulations are often found to be cumbersome when the polynomial representation is used. The matrix formulation simplifies the analysis.

Second, the diagnostic properties of a signature analyzer are revealed by this approach [64].

In this formulation, the content of an m-bit SAR at any cycle t is represented by an m-dimensional row vector,

$$S_t = \begin{bmatrix} s_{t,0} & s_{t,1} & s_{t,2} & \cdots & s_{t,m-2} & s_{t,m-1} \end{bmatrix} . \tag{2.29}$$

A SAR with a characteristic polynomial $f(x)$ can be represented by an $m \times m$ state transition matrix C, where

$$C = \begin{bmatrix} c_0 & 1 & 0 & 0 & \cdots & 0 \\ c_1 & 0 & 1 & 0 & \cdots & 0 \\ c_2 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{m-2} & 0 & 0 & 0 & \cdots & 1 \\ c_{m-1} & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{2.30}$$

for a Type-1 analyzer, and

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ c_{m-1} & c_{m-2} & c_{m-3} & c_{m-4} & \cdots & c_0 \end{bmatrix} \tag{2.31}$$

for an equivalent Type-2 register.

Comparing a MISR to an LFSR having the same characteristic polynomial, the only difference between them is in the response vector R_t, where

$$R_t = \begin{bmatrix} r_{t,0} & r_{t,1} & r_{t,2} & \cdots & r_{t,m-2} & r_{t,m-1} \end{bmatrix} \tag{2.32}$$

for a MISR, and

$$R_t = \begin{bmatrix} r_{t,0} & 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{2.33}$$

for an LFSR.

The recursive relation of the SAR can be described by the vector equation:

$$S_t = S_{t-1}C + R_t \ . \tag{2.34}$$

If the initial state of the SAR is S_{-1}, then the above recursion formula can be expanded to find the t^{th} value of the signature, as shown in the following equations:

$$\begin{aligned}
S_0 &= S_{-1}C + R_0 && (2.35)\\
S_1 &= S_0C + R_1 && (2.36)\\
&= S_{-1}C^2 + R_0C + R_1 && (2.37)\\
S_2 &= S_1C + R_2 && (2.38)\\
&= S_{-1}C^3 + R_0C^2 + R_1C + R_2 && (2.39)\\
&\vdots\\
S_t &= S_{-1}C^{t+1} + \sum_{i=0}^{t} R_iC^{t-i} \ . && (2.40)
\end{aligned}$$

The final signature, after compacting all t MUT responses ($0 \le t < T$), is given by:

$$\begin{aligned}
S &= S_{T-1} && (2.41)\\
&= S_{-1}C^T + \sum_{t=0}^{T-1} R_tC^{T-1-t} \ . && (2.42)
\end{aligned}$$

Based on Equation 2.42, the following observations can be made regarding the final signature [64] :

1. The decomposition between the initial SAR state "S_{-1}" and the single-cycle signatures "S_t". Specifically, after T-cycles of compaction,
   - the initial-state signature is $S_{-1}C^T$, and
   - the signature at cycle "t" is $S_t = R_tC^{T-1-t}$.
2. The relation between "S_t" and "t" is characterized by C^{T-1-t}, the power of the transition matrix. If $k = T-1-t$, then the computation of the single cycle syndrome S_t reduces to computing C^k.

The matrix formulation can be directly extended to the error domain. Let $E_t = R_t \oplus R_t^0$ be an m-bit row vector representing the error(s) at cycle "t". Then the single cycle error

signature is characterized by:

$$S_{e,t} = E_t C^{T-1-t} \ . \tag{2.43}$$

In general, if the SAR is initialized to zero before compaction, then the final error signature is the bit-wise modulo-2 summation of the syndromes of each failing cycle [64], or

$$S_e = S_{e,i} \oplus S_{e,j} \oplus \cdots S_{e,l} \ . \tag{2.44}$$

Finally, it should be mentioned that a more compact matrix formulation can be obtained in GF(2^m) [94]. In GF(2^m), a signature analyzer multiplies its contents by α and adds it to its input R_t, where α is an element of GF(2^m) corresponding to the root of the feedback polynomial [94].

When the last response is compacted, the contents of the SAR is the *time* signature

$$S = S_{-1}\alpha^T + \sum_{t=0}^{T-1} R_t \alpha^{T-1-t} \ . \tag{2.45}$$

If the SAR is cleared before compaction, then $S_{-1} = 0$ and,

$$S = RH_t \ , \tag{2.46}$$

where

$$R = \begin{bmatrix} R_0 & R_1 & R_2 & \cdots & R_{T-1} \end{bmatrix} \tag{2.47}$$

is the output response vector in GF(2^m), and

$$H_t = \begin{bmatrix} \alpha^{T-1} \\ \alpha^{T-2} \\ \vdots \\ \alpha \\ 1 \end{bmatrix} \tag{2.48}$$

is the *time compaction matrix*. It should be mentioned that H_t is the check matrix of the $[T, T-1, 2]$ *Reed-Solomon* code over GF(2^m) [84].

2.5.7 Aliasing In Signature Analysis

Fault detection occurs if the signature realized by a module differs from that of a fault-free copy of the module. However, since the volume of test response data in a SA is reduced by orders of magnitude, a loss of information may occur resulting in $S_e = S^0$. This in turn translates into the possibility of a faulty module being declared fault-free. This masking problem is termed *aliasing* [1].

For any BIST compaction scheme, it is crucial to have a measure of the possible aliasing [95]. The problem of aliasing is generally characterized by its probability of occurrence. Minimizing the aliasing probability is a good measure of the viability of any compaction technique.

In order to calculate this probability, it is sufficient to consider only the error sequence $E(x)$ [1]. Due to the linearity of signature analyzers, the aliasing analysis can be limited to just considering a compactor initialized to zero, with the error sequence being its input source. With this assumption, aliasing will occur if the analyzer returns to the all zero state [96]. This will happen if and only if the error polynomial is divisible by the feedback polynomial [97].

It is known from the subject of coding theory that the set of all 2^m-ary $E(x)$ polynomials which are divisible by $G(x)$ constitutes a code with a generator polynomial $G(x)$ [98]. This code is called the Aliasing Code (AC) of the compactor [89, 99]. Thus, the problem of aliasing is equivalent to the problem of finding the probability of undetected errors in the aliasing code corresponding to the signature analyzer [100].

With the introduction of SA techniques for testing boards [83], the probability of aliasing has been the subject of extensive research [96, 101, 102, 103, 104, 105]. Several papers have been published using various error models [95]. Error models correlate the fault model used with the module with the error patterns observed at the module output(s) [97]. Measuring aliasing probabilities in the error domain yields results very close to those obtained from experiments performed in the fault domain [105].

Several alternatives have been proposed for computing such probabilities, ranging from computationally intensive exact solutions, to bounds, and approximate heuristic solutions [95].

The most accepted result is that the final steady state value of the aliasing probability is equal to 2^{-m} for an m-bit LFSR [101]. However, it has been established in [96] that

primitive polynomials converge much faster to this final steady state value, and have in general smaller aliasing probabilities than non-primitive polynomials of the same order. The performance of a non-primitive polynomial may be just as good as the primitive one in some cases, but there are many more cases in which it is much worse.

Both LFSR structures (Type-1 and Type-2) have the same masking characteristics, since an error polynomial $E(x)$ will cause aliasing if and only if $E(x)$ is a multiple of $G(x)$ [1].

An investigation of the properties of an m-bit MISR also shows that the steady state value of the aliasing probability is equal to 2^{-m} [103]. Similar to LFSRs, some unique features of primitive MISRs have been proven [103].

First, it has been shown that they provide a better-performance by minimizing aliasing [103]. They guarantee the aliasing probability limit of 2^{-m}, while counter examples show that non-primitive MISRs cannot. Their performance is not affected by permutations of MISR inputs, so that any MUT output can be connected to any MISR input without affecting its aliasing characteristics [103].

Furthermore, the aliasing probability for large test lengths is the *same* under all error models, no matter *which* primitive feedback polynomial is used [100].

The Aliasing Effect on Test Length and Diagnosis

Aliasing has a direct impact on the detection confidence and the test length required to achieve an acceptable fault coverage. In addition, aliasing has a direct impact on any diagnostic procedure that uses signature analysis. This is simply because if a fault cannot be detected in the first place, then there is no hope of diagnosing it.

Hence, care must be devoted to the proper selection of feedback polynomials that minimize the aliasing probability and increase the detection confidence after compaction. The appropriate pseudorandom test length must also be determined to achieve the above objectives [1, 104, 106].

Henceforth, it will be assumed that the signature analyzers are selected to minimize the aliasing effect, and the pseudorandom test length T is sufficient to provide the required fault coverage.

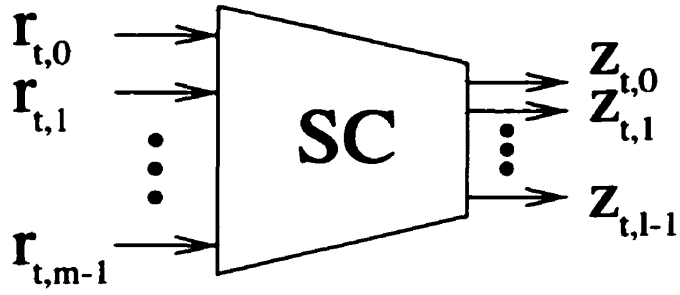


Figure 2.13: An m-input l-output space compactor.

2.6 Space Compaction In BIST

Off-line testing of large VLSI modules with many outputs requires a large time compactor and a large memory for reference signature data storage. It is reasonable to assume that not all m outputs will be erroneous for any given test [107]. Thus, there is no need to monitor the error free outputs. However, the fault sights, i.e., the outputs for which the fault causes the values observed to differ from the fault free ones, are not known a-priory.

The number of erroneous outputs at any given test depends on the number of faults and the module structure. Since the number of tests applied in BIST is large, there is a reasonable probability that a fault will manifest itself as an erroneous response on several of the m outputs in several tests. Thus, it seems that it is possible to reduce the number of monitored outputs during test while retaining the requirement of propagating at least a single erroneous bit from the MUT to the monitored outputs. This is the basic principle of *space* compaction.

A space compactor, such as the one shown in Figure 2.13, is a combinational circuit that accepts the m outputs of a MUT as inputs and provides a reduced number of outputs (l) for monitoring while testing the MUT [107]. The space compaction factor is thus equal to $\frac{m}{l}$. Clearly the objective is to minimize l. A considerable reduction in the test response width is then possible.

Space compaction in BIST is usually combined with time compaction to provide an optimal compression of test responses [8, 38, 107, 108]. Field compaction from $GF(2^m) \rightarrow GF(2^l)$ allows the use of an l-stage SAR instead of the m-stage one that would have to be used if the space compactor was not employed. This reduces the hardware overhead required by the SAR and the reference signature data storage, which in turn offsets the additional hardware required by the space compactor.

A space compactor is implemented as the $(m \times l)$ matrix

$$H_s = \begin{bmatrix} h_{0,0} & h_{0,1} & \cdots & h_{0,l-1} \\ h_{1,0} & h_{1,1} & \cdots & h_{1,l-1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{m-1,0} & h_{m-1,1} & \cdots & h_{m-1,l-1} \end{bmatrix} , \tag{2.49}$$

where $h_{i,j} \in \mathrm{GF}(2)$. It should be mentioned that H_s is the check matrix of an error detecting-correcting code [84, 107, 109]. The space-compacted output (Z_t) corresponding to the t^{th} test is related to R_t through the relation:

$$Z_t = R_t H_s , \tag{2.50}$$

where,

$$Z_t = \begin{bmatrix} z_{t,0} & z_{t,1} & \cdots & z_{t,l-2} & z_{t,l-1} \end{bmatrix} . \tag{2.51}$$

The major problem in designing space compactors is that of minimizing the number of outputs "l" for a given MUT. Several codes have been used in designing these compactors, but linear Hamming codes [84] have been among the most popular ones.

The design of an optimal $\mathrm{GF}(2^m) \rightarrow \mathrm{GF}(2^l)$ space compactor is presented in [6]. Lower bounds on l are provided in [107], where it is also concluded that space compaction based on non-binary codes, such as Reed Solomon codes [84], offer additional diagnostic advantages.

Finally, a word about the masking effect in space compaction. This occurs when the error patterns are not propagated to the outputs of the compactor. It has been shown in [109] that the portion of errors not propagated by an m-input l-output compactor is less than 2^{-l}. If i is the minimum number of different error patterns caused by a fault during BIST, then the probability of not propagating errors is 2^{-il} [109]. As mentioned in Section 2.5.7, this has a direct impact on the diagnosis process, because if a fault is not propagated, then it can neither be detected nor diagnosed.

Chapter 3

Chip-Level Diagnosis

Signature analysis has been an integral part of BIST for the past seventeen years. It has been traditionally used in testing as an effective method for determining the pass/fail status of a MUT. Once the presence of a fault is detected, a diagnostic routine is often invoked to identify the cause(s) of the failing MUT so that a corrective action can be taken.

It is very desirable to diagnose the fault(s) based on the information of the faulty signature [110]. However, compared to fault diagnosis in non-BIST environments, BIST fault diagnosis is more difficult [1].

One diagnostic method in common use is fault simulation, which identifies the fault equivalence class(es) that explain(s) the observed erroneous behavior. This classical diagnostic method begins by determining the failing tests, which usually requires a good-machine simulation of the chip to compute its expected response for each test, followed by a test by test comparison with the observed behavior of the faulty MUT.

However, during BIST, the only failure data available is the final erroneous signature. Thus, the individual responses cannot be determined to be faulty or not, and the complete failure data so necessary for a diagnostic fault simulation, is no longer available.

Furthermore, since the test patterns are generated pseudorandomly by LFSRs or MISRs, there is no pre-defined test set such as the one required with conventional diagnosis, which explicitly defines the chosen diagnostic test patterns.

An obvious approach to alleviating this problem is exhaustive test matching where the response of each test pattern is logged-out and bit-wise compared with its expected value from simulation. The number of test patterns required to isolate the error location(s) by

such a method is always equal to the length of the pseudorandom test session.

However, this approach has been found to be costly in practice as it requires unloading the MUT's response for each test, a procedure that is both complex in implementation and time consuming in practice [64, 72, 110].

In addition, this approach also requires the storage of a complete fault dictionary. Since the number of applied pseudorandom test patterns is usually very large, to ensure a high fault coverage, storing all of the applied tests and their expected MUT responses requires a large amount of memory and computation time.

The size of these dictionaries will be so huge that it negates the benefits of BIST compaction and pseudorandom testing in the first place [64, 72, 110]. Hence, the traditional diagnostic techniques using pre-calculated fault dictionaries are ineffective, inefficient, and simply not practical for large structures such as VLSI chips in a BIST environment.

Without test patterns to simulate and output responses to match against the results from fault simulation, a signature-based BIST fault diagnosis faces different and more difficult challenges than those with conventional non-BIST diagnostic testing [18].

Motivated by these challenges, signature analysis found significant applications in BIST fault diagnosis based on the faulty signatures.

In performing diagnosis, if the fault(s) cannot be located directly from the faulty signature, then it is necessary to identify some or preferably all of the failing tests from the compacted signature [72]. Given the failing tests, the traditional diagnostic simulation practices, used with non-BIST circuits, can be employed to isolate the possible faults.

Each failing test identifies a set of candidate faults. A number of failing tests narrows the search space to the intersection of their candidate faults. Obviously, it is important to identify as many failing tests as possible [111]. The more complete the list of failing tests, the smaller the class of faults whose behavior can explain the observed failures, and thus the higher the diagnostic resolution.

A failing test is identified by locating the erroneous bit(s) in the compacted TRS corresponding to the applied test sequence. The complete identification of all erroneous bits in the test response sequence is confounded by the fact that signature analysis is a compaction technique as opposed to a lossless compression technique, and also by traditional signature aliasing.

Identifying the erroneous bits entering the signature analyzer has been the subject of

research for the past decade. Several researchers have reported their progress in this field. Analytical diagnostic techniques for both single-input and multiple-input analyzers have been presented.

Although these techniques guarantee the correct identification of errors, they suffer from a major deficiency by assuming few erroneous bits in the test response sequence. This assumption is generally unrealistic, since even a single defect in a MUT can cause hundreds or even thousands of errors in the response sequence [38]. Consequently, the results obtained from these techniques are often misleading.

Another class of BIST fault diagnosis techniques was developed based on post-test fault simulation. Compared to the analytical techniques, the post-test simulation ones can usually provide better diagnostic resolution, since they utilize more information of the faulty MUT. The major deficiency of these techniques is their large memory and tester requirements.

In the rest of this chapter, a comprehensive survey of these SA-based BIST diagnostic techniques will be presented, emphasizing their applicability, practicality, pros, cones, and cost.

3.1 Locating the Faults

A structured DFT design permits the use of fast fault simulation, making it feasible to simulate faults for diagnostic testing [112]. Strictly speaking, a faulty signature does not provide sufficient diagnostic information to be able to perform any diagnostic procedure.

Without test patterns to simulate and output responses to match against the fault simulation results, BIST fault diagnosis faces different challenges from those facing conventional non-BIST diagnosis [18]. Hence, it seems that the diagnosis of these circuits is simply *impossible*. However, there is actually *"hope"*. To meet these challenges, both *pre-test* and *post-test* fault simulation methods have been suggested in [5, 18, 112, 113, 114, 115, 116].

In the remaining of this section, a comprehensive survey of these simulation-based techniques is presented.

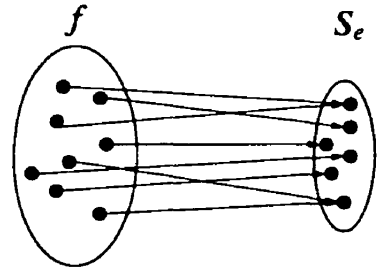


Figure 3.1: Mapping the fault domain to the signature domain [4].

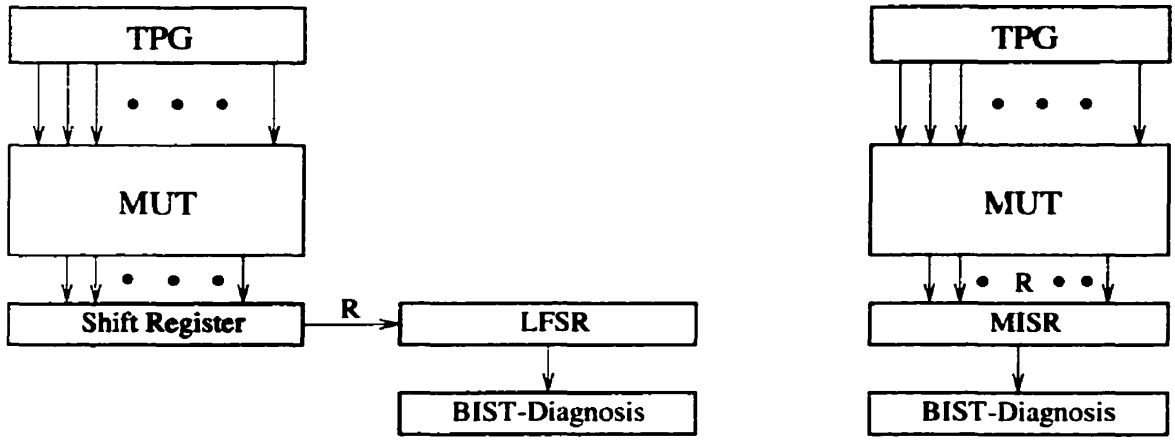


Figure 3.2: Test models for chip-level BIST fault diagnosis.

3.1.1 BIST Fault Diagnosis With Pre-Test Simulation

It is obvious that an error, observed in a faulty output response, is a manifestation of the fault(s) in the MUT's hardware. Therefore, an error signature S_e is associated with some corresponding fault(s) in the MUT. The process of BIST testing with signature analysis can thus be viewed as a mapping between the set of faults and the domain of signatures as shown in Figure 3.1 [4].

Assume that the MUT is tested as in Figure 3.2. In Figure 3.2(a), the output responses from a multiple-output combinational MUT are shifted serially into the SAR, implemented as an LFSR, with no additional area overhead other than the shift register and the usual BIST circuitry.

The test response sequence R is a concatenation of m-bit blocks, each block corre-

sponding to the m-outputs of the MUT for a given test. These blocks are shifted out completely from the output register before a new test pattern can be applied. Hence, R can be considered as being the output response of a single output MUT. When the MUT actually has a single output, the shift register is not required. Finally, if the MUT contains sequentials implementing a single scan-chain design, the shift register corresponds to the scan register implemented on the MUT.

In Figure 3.2(b), the output responses from a multiple-output combinational MUT, or a MUT implementing a multiple scan-chain design, are shifted in parallel into the SAR implemented as a MISR.

The objective here is to be able to locate the fault(s) using only the observed final error signature S_e. Without modifying the MUT, the only straightforward solution is to create, prior to testing, a dictionary of faulty signatures corresponding to each of the f faults in the list associated with the MUT [4, 105, 117].

The signature dictionary is created using a fault simulation, which models the effect of each f on the MUT's primary and/or pseudo outputs. A signature generation program uses the simulator's predictions about the faulted MUT output states to compact a faulty signature in a simulated SAR. Note that nothing has been assumed thus far about the employed fault model or the multiplicity of faults. These will determine the efficiency, complexity and the type of simulation to be performed. For example, if the single stuck-at model is used, then the signatures can be obtained using the PPSFP fault simulator [18].

The diagnostic procedure can be summarized as follows:

1. Execute a BIST session and obtain the compacted signature.
2. Compare the above signature with the pre-computed reference signature. If they match then no faults are detected and no diagnosis is required.
3. If the comparison result is negative (i.e., they are different), then some fault(s) is/are present. Diagnosis is done off-line with a lookup program which compares the observed faulty signature with each of the f signatures in the fault dictionary. For each match, the corresponding fault is added to the final list of candidate faults.

After performing diagnosis, the list of candidate faults may contain a single fault. This is usually the case when the corresponding fault produces a unique signature. In this case the highest diagnostic resolution is obtained.

The list of candidate faults may also contain multiple faults. This is usually the case when these faults have identical signatures. Hence, in this case the diagnostic resolution is down to a single class of indistinguishable faults. It should be noted that these faults are not distinguishable *signature-wise*. This does not necessarily imply that they belong to a class of indistinguishable faults under the assumed fault model. For example, if the size of the SAR is increased, then these faults may have different signatures.

Finally, the list of candidate faults may be empty. In this case no faults are identified. This may happen if the actual fault was not modeled, or the employed fault model was not adequate to model the actual physical faults that may have occurred in the MUT. If this is the case, then the fault list must be expanded to include additional faults or a different fault model may be used. A mix of fault models may also be used.

The Diagnostic Resolution of a SAR

It is clear from the above analysis that for best diagnostic resolution, the number of distinguishable signatures must be maximized. A fault can be uniquely identified provided that it produces a unique signature that cannot be generated by any other fault [4]. Thus for maximum diagnostic resolution, the number of distinguishable signatures must be f, i.e., a different signature for each of the modeled faults.

In general, only equivalent faults are supposed to have the same signature. But unfortunately, a lot of the error information is lost due to the compaction process, thus breaking the one-to-one mapping between the error sequences and the faulty signatures. Therefore, two non-equivalent faults may also produce the same faulty signature.

Misdiagnosis results in unnecessary corrective actions that might be costly. Hence, for this diagnostic approach to be viable, it is crucial to minimize the probability of misdiagnosis. The probability of misdiagnosis is defined as the probability of any pair of distinguishable faults producing the same faulty signature [100]. Since this definition is analogous to the signature aliasing concept, the probability of misdiagnosis will be termed the probability of *Diagnostic Aliasing* (P_{DA}).

The diagnostic aliasing problem was addressed in detail in [4, 105, 117]. A theoretical model was developed that relates the average number of faults generating unique signatures to the number of stages (m) in an LFSR and the size of the fault list (f) associated with the MUT. In this model, it is assumed that:

1. The applied pseudorandom test set is sufficiently large to detect all the modeled faults.

2. Each fault can generate any of the $2^m - 1$ possible signatures with the same probability, where one signature corresponds to the fault-free response. Furthermore, the signatures generated by any pair of faults are statistically independent.

Since the model is quite general, it can be used to analyze any BIST compaction scheme that satisfies these assumptions. The relation between m, f, and the number of faults which are signature-wise indistinguishable (f_{si}) was captured by the following approximation:

$$m \approx \log_2 \frac{f^2}{f_{si}} \tag{3.1}$$

$$= 2\log_2 f - \log_2 f_{si} \ . \tag{3.2}$$

The above relation was verified with simulation experiments on the ISCAS-85 benchmark circuits, using a gate level simulator and LFSRs implementing both primitive and non-primitive polynomials [4, 105, 117]. It should be noted that these experiments focused on the identification of the stuck-at faults in the MUT and were, therefore, a function of the specific MUT. However, similar results were obtained for all circuits and hence this simple relation holds for all MUTs satisfying the model assumptions.

The following was concluded from both the simulation results and the above formula:

1. In order to obtain a certain level of diagnostic resolution, the required LFSR must have the space of possible signatures approximately 1000 times larger than the size of the fault list.

2. The average number of different signatures and the average number of uniquely generated signatures are significantly less in LFSRs implementing non-primitive polynomials than those implementing primitive polynomials. This is due to the long initial transient period in non-primitive polynomials.

3. As the size of an LFSR increases by 1, the percentage of faults which cannot be uniquely identified drops approximately by half. Figure 3.3 clearly shows the advantage of employing large LFSRs in decreasing the probability of diagnostic aliasing.

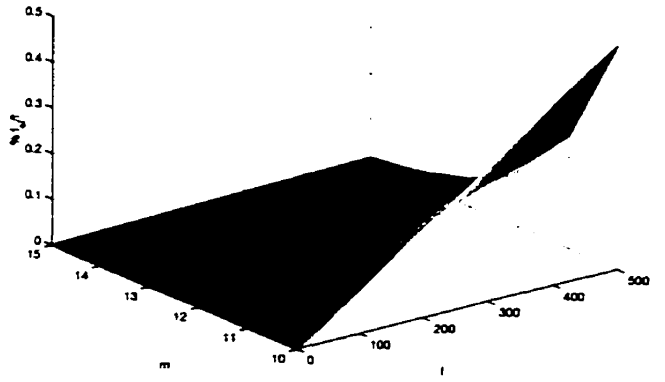


Figure 3.3: Percentage of signature-wise indistinguishable faults.

4. For a given size of the LFSR, as the total number of faults doubles, the percentage of faults which cannot be uniquely diagnosed increases approximately twice as much. Figure 3.3 shows the degradation in the diagnostic resolution due to the increasing number of faults in a MUT for a fixed size LFSR.

5. The probability of misdiagnosis or diagnostic aliasing can be approximated from Equation 3.1 as follows. Let

$$f_{si} = p \times f \ , \tag{3.3}$$

where p is the percentage of unresolved faults. Substituting Equation 3.3 into Equation 3.1 yields:

$$m \approx \log_2 \frac{f^2}{p \times f} \tag{3.4}$$

$$= \log_2 \frac{f}{p} \ . \tag{3.5}$$

Rearranging Equation 3.5, and noting that $\frac{1}{f}$ is the probability of a fault, assuming all faults are equally likely and statistically independent, then

$$2^{-m} = \frac{p}{f} \tag{3.6}$$

$$= p \times \frac{1}{f} \tag{3.7}$$

$$= P_{DA} \ . \tag{3.8}$$

Thus, the probability of diagnostic aliasing is quite low. This indicates that signature dictionaries can be quite effective as there is little chance of misdiagnosing faults. Although this result was derived for an LFSR, it was shown in [100] that MISRs of size $m \geq 6$, also have $P_{DA} \approx 2^{-m}$. Hence, the result can be generalized to any SAR satisfying the model assumptions.

These simple results can be employed directly to assess the effectiveness of this diagnostic technique in BIST'ed VLSI circuits. For a given fault list and the desired diagnostic resolution, the size of the SAR which guarantees such resolution can be easily calculated along with the probability of misdiagnosis.

In conclusion, fault diagnosis using a dictionary of faulty signatures and a look-up program can be easily implemented in software with no additional area overhead. Diagnosis is only possible when there is an exact match between a dictionary signature and the compacted one.

The major drawback of this diagnostic method is the large amount of pre-test simulation needed to compile the reference faulty signatures. The simulation effort is proportional to the size of the fault list. The size of the dictionary database should be minimized to conserve memory resources and shorten the analysis time on the test system.

Another drawback of this approach is that it requires the use of large size SARs to enhance the diagnostic resolution. Finding primitive polynomials for large compactors is not an easy task. In addition, these rather expensive compactors may present serious implementation problems.

Finally, it should be mentioned that this diagnostic technique has been extended in [51] to obtain unambiguous diagnosis of the faulty partitions in the MUT as opposed to the actual faults in it. A Programmable Cellular Automata (PCA) test-bed is configured as a maximum-length, group CA for test response compaction and subsequently as a non-group CA for classifying the faulty signatures into disjoint groups that map into different partitions of the MUT.

3.1.2 BIST Fault Diagnosis With Post-Test Simulation

The basic idea here is to obtain a small yet sufficient failing response sample to a known stimuli, and then use fault simulation to determine the fault equivalence class(es) that can

cause the observed behavior. Usually, several stimulus/faulty-response pairs are needed to resolve a fault to a single fault equivalence class [1].

As mentioned at the beginning of the chapter, a pseudorandom-pattern diagnostic simulation costs more than the one with deterministic testing patterns. This is because of the added step of simulating the pseudorandom TPG to create the test patterns to be simulated. In addition, pseudorandom test patterns are in general much longer than their deterministic counterparts in order to achieve an acceptable fault coverage. Consequently, the simulation effort involved in creating the fault dictionaries may become quite excessive [5].

Nevertheless, there are diagnostic advantages in testing with pseudorandom test patterns. Since a pseudorandom test generally applies many times more patterns than a deterministic test, most faults are detected many times more as well [112]. With these extra patterns, there is a greater probability of including test patterns that are useful for diagnosis but redundant for fault detection purposes.

In addition, these extra patterns significantly improve the diagnostic resolution for defects that do not behave exactly as stuck-at faults [112]. Since only the failing patterns can be used to resolve the faults associated with these defects, the extra failing patterns provide additional opportunities to eliminate *bogus* fault candidates [112].

In the remaining of this section, two post-test diagnostic simulation techniques will be presented.

BIST Diagnostics With Intermediate Signatures

BIST fault diagnosis based on post-test simulation techniques with intermediate signatures have been reported in [1, 18, 112, 114, 115, 116, 118]. The basic idea is to obtain a failing response to a known stimuli and then use post-test fault simulation, at intermediate points in the test sequence, to determine the fault equivalence class(es) that could have caused the observed response. Typically, several stimulus/faulty-response pairs are required to resolve a fault to a single fault equivalence class [1].

The MUT is assumed to be designed according to the LSSD guide lines. The general structure of the diagnostic setup has the form shown in Figure 3.4. The LSSD latches are treated as pseudo-primary inputs/outputs of the combinational logic connected to them.

This method relies on some highly efficient fault simulators, such as the Parallel Pattern

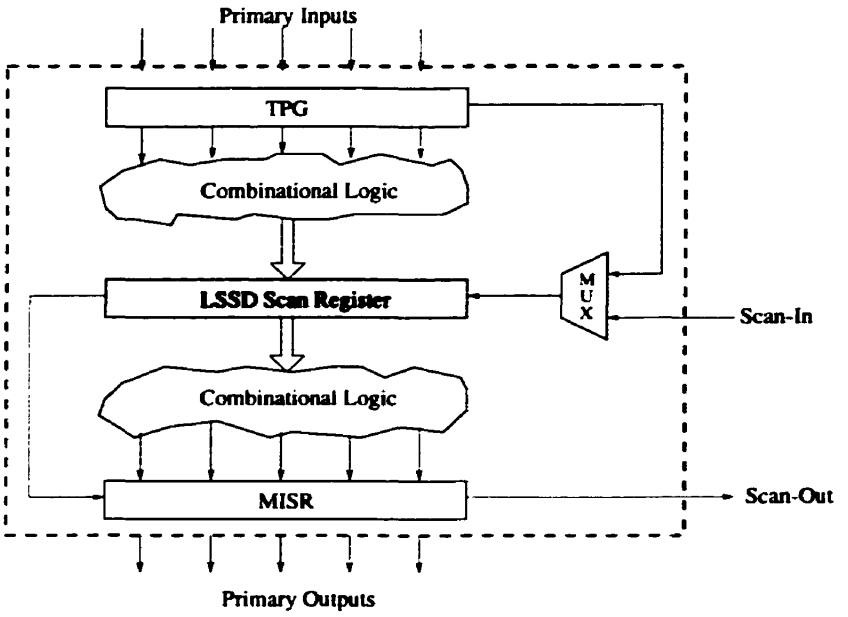


Figure 3.4: BIST fault diagnosis using intermediate signatures.

Single Fault Propagation (PPSFP) fault simulator [18, 112], that have been modified to be suitable for diagnostic simulations in BIST environments. These simulators make use of the fact that all LSSD latches are both observable and controllable, by performing only two-valued combinational logic simulations.

A test system is required to support this simulation-based diagnostic technique [1]. Since the total number of pseudorandom test patterns is usually large, then it makes sense to break down the application of tests to groups, called *procedures* [118], and check the compacted signature at the end of each group for errors. This is necessary to reduce the volume of test data that must be collected by the tester, and minimize the complexity of the fault simulator.

The primary added data requirement is a dictionary of intermediate signatures. Prior to testing, the expected good-machine signatures are pre-calculated for all T_p-pattern procedures. These signatures are available for real-time comparison at the tester. The tester can then immediately isolate a failure to a set of T_p patterns.

The value of T_p is usually chosen to be equal to the number of patterns that can be simulated in parallel by the post test simulator. In [18, 112, 118] the signatures are collected at intervals of 256 LSSD patterns, because the PPSFP simulator can simulate one byte of patterns in parallel. In [1, 115], each procedure consists of 100 test patterns.

The test system executes the following for each test procedure until a failing one is observed:

- Initialize the MUT. This includes placing the proper seeds in the TPG and MISR, and setting all SRSs and memory blocks to known states. The TPG seed at the beginning of each procedure is also saved.
- Supply the MUT with clock sequences sufficient to apply T_p test patterns.
- Unload the intermediate signature from the MISR.
- Compare the unloaded signature from the MUT with its reference in the signature dictionary.
- If the MUT displays the correct signature, then the test system restores its state to what it was at the completion of the current test procedure before the signature was scanned out, and the next procedure is initiated.

If an intermediate signature differs from its corresponding reference in the dictionary, then at least one failing response occurred in the most recent test procedure. Such signature is noted, and the corresponding procedure is declared as a failing one.

It is desirable to perform diagnosis based on the results of only a few failing procedures [118]. This is usually sufficient, since each procedure provides T_p patterns worth of passing and failing information. The diagnostic algorithm collects the MUT data for a small sample of failing patterns within those procedures. The following steps are used in obtaining this sample of data:

1. Upon the determination of a failing procedure, the test system restores the MUT state to the starting point of this most recent procedure.
2. The failing test procedure is re-applied to collect the test response data at the primary outputs and the SRS contents of the MUT. This is done by configuring the MISR as a simple shift register. So instead of compacting the responses, they are scanned out through the rightmost bit in the MISR, and saved in the test system. Thus, only the data necessary for diagnosis is collected.

3. Since the TPG contents at the beginning of any procedure are stored in the test system, the T_p patterns in the current failing procedure can be regenerated by a functional simulation of the TPG, with these values as its initial seed.

4. A failing procedure causes the state of the MUT to deviate from its supposed fault-free one. This will in turn result in an incorrect state for all subsequent procedures regardless of whether they capture errors or not. Hence, the test system must restore the state of the MUT to that of a good part at the end of each failing procedure. This includes reseting the MISR to the correct value. In [116], the MISR value is not restored and future test procedures are salvaged by adjusting their expected reference signatures to fit the previously observed errors. Thus, the diagnosis can continue and additional failure data can be collected from other procedures. This additional data is used to improve the diagnostic resolution and provide the capability to diagnose multiple defects.

When the collected failure data is sufficient for a diagnostic analysis, the MUT is removed from the tester thus allowing it to continue testing other modules [112]. The remainder of the diagnosis is done off-line using well-known deterministic simulation techniques. The diagnostic analysis consists of the following steps starting with the first failing procedure and proceeding to the last one:

1. Determine the T_p patterns to be simulated with the *good machine*, corresponding to the failing procedure under consideration. The simulator input is the collected failure data, the MUT's logic model, and the BIST session data such as TPG seeds and clocking order. The simulation goal is to determine the gates, nets, and logic macros containing the fault(s) [118].

2. Perform a two-valued good machine simulation using the above patterns on a parallel-pattern simulator. The simulator calculates the fault-free values of the primary outputs, SRSs, and the internal logic macros.

3. Compare the calculated values to the corresponding data collected by the tester to determine the stimulus/faulty-response pairs in the test procedure. Obviously, not all patterns will expose a fault.

4. Using the first failing-pattern data and the corresponding good-machine data, an *initial candidate fault list* can be obtained by analyzing the MUT structure. In

determining the fault candidates, a hierarchy of fault models can be used which includes single stuck-at faults, multiple stuck-at faults, RAM faults, and open or shorted nets [118]. Further simulation is avoided if the fault list includes a single candidate.

5. Finally, perform a faulty-machine simulation on the identified failing patterns by injecting faults from the candidate list. This will identify the faults which can explain the observed values during test. According to the selected *fault reduction* mode [18, 112], candidate faults in the initial list can be eliminated, and only those explaining the observed data in all the preceding procedures are retained.

Once the diagnostic analysis is complete, the list of potential faults is available. This list contains faults that continuously explain the failures in the failing procedures and are thus included in the final diagnostic results. The simulator qualifies each potential fault with a confidence level. This indicates whether the faulty-machine simulation matched all the observed passing values, matched all the observed failing values, or any combination of the two. The highest confidence is obtained when both values match, and the lowest when neither does [118].

In conclusion, post-test simulation with intermediate signatures makes the use of simulation for the purpose of diagnosis practical. Moreover, by simulating faults after test, the simulation can be optimized for a specific failure, resulting in a highly efficient diagnosis. In general, it is possible to diagnose a defect to a single fault-equivalence class [112].

One consideration that must be taken in any diagnostic procedure utilizing BIST is the possibility of signature aliasing. In this case, some faults maybe missed. Clearly, the longer the test stream whose output is compacted, the greater the probability of aliasing. A byproduct of subdividing the test process is that the probability of aliasing is reduced [119], thus increasing the effective coverage of the diagnostic process.

This method also has the advantage of not requiring the MUT to be modified for diagnostic purposes. This hardware saving is offset by the fact that this technique requires the use of an external tester for on-line decision making, i.e., it decides on the diagnostic data to be collected based on the intermediate signatures. The external tester in-turn slows down the whole diagnostic process because of the data transfer overhead between the MUT and the tester.

Despite the fact that this diagnostic strategy minimizes the tester effort by collecting

failure data from a normal test application [112], the large tester memory requirement is a major deficiency of this technique.

Finally, a variety of methods have been suggested in [18, 112] to cut down on the number of faults to be simulated. Still, the fault dictionary remains large, and a full circuit simulation is required to obtain each entry in the dictionary. Since pseudorandom tests are in general much longer than their deterministic counterparts, the required amount of fault simulation can be excessive. This may gradually negate the cost savings justifying data compaction in BIST.

BIST Diagnostics Without Intermediate Signatures

Another approach to single fault identification, using signatures compacted by a MISR, was developed in [5]. The diagnostic method, named DAPPER, is designed to work with combinational multiple-output modules but can be easily extended to modules employing an LSSD scan design.

DAPPER focuses on the identification of single stuck-at faults in the MUT, and is therefore a function of the specific MUT. It partitions the entire fault list into *classes* based on the fault's detection probability, deduced from the observed behavior of the MUT. This provides an initial coarse resolution by dropping a large number of faults from the fault list without performing any simulation, thus eliminating the need for collecting intermediate signatures.

The diagnostic setup used with DAPPER is shown in Figure 3.5. It is structurally similar to the "output data modification" scheme proposed in [120].

Since a MISR is used to compact the MUT outputs, then the fault detection probabilities must be calculated from it. It is suggested in [5] that the quotient stream be analyzed to determine these probabilities. With a conventional MISR, however, all faults tend to have the same detection probability of $\frac{1}{2}$ on the quotient stream [5]. In order to preserve the detection probabilities through the MISR, the feedback connection is removed thus degenerating the MISR into a MINFSR.

In a MINFSR the contents of each SRS is the XOR of the incoming test response bit, with probability $P_R[i]$, and the output of the previous SRS which has probability $P_{SRS}[i-1]$. It can be easily shown using a Venn diagram that the signal probability $P_{SRS}[i]$ is equal

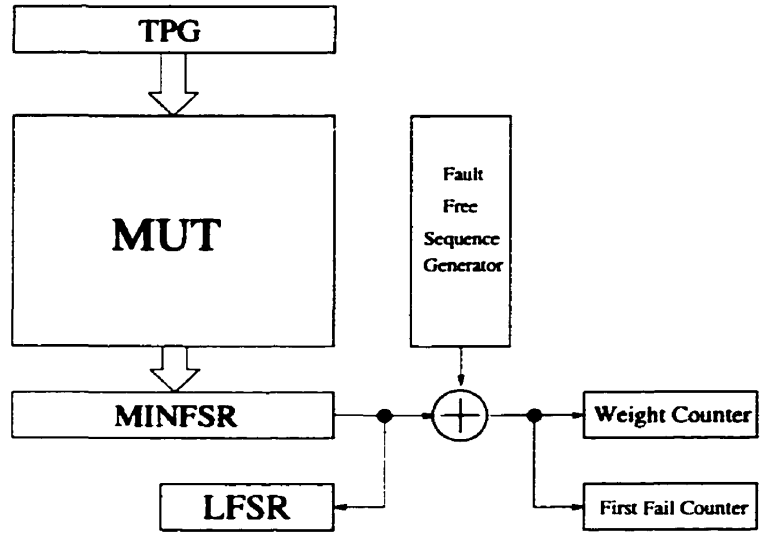


Figure 3.5: The DAPPER diagnostic setup [5].

to:

$$P_{SRS}[i] = P_R[i] + P_{SRS}[i-1] - 2P_R[i]P_{SRS}[i-1] \ , \tag{3.9}$$

where $0 \leq i < m$, and $P_{SRS}[-1] = 0$. Hence the detection probabilities can be easily calculated. Once the detection probabilities for all m outputs are available, then the quotient bit detection probability can be calculated for each fault using fault simulation.

Since the detection probabilities are used as a distinguishing attribute for the faults in the MUT, then the fault detection probability must also be determined from the observed quotient stream in the MUT. This is done by converting the stream to an error sequence. The T-bit fault-free quotient stream available from a fault-free simulation is stored in a RAM constituting the error-free sequence generator. Bitwise XORing the error-free quotient stream with the observed one produces the error sequence associated with the existing fault(s).

Measuring the weight of the error stream (w_f) provides an estimate of the detection probability for any fault which may have occurred. Thus,

$$p_{f_m} = \frac{w_f}{T} \ , \tag{3.10}$$

where p_{f_m} denotes the measured fault detection probability. In order to measure w_f, the error sequence is used to increment a modulo-T binary counter as shown in Figure 3.5. The counter is cleared at the beginning of the test. The value of the counter at the end

of the BIST session is equal to w_f. Clearly, the accuracy of the estimated fault detection probability increases with test length [5].

Since several faults may have similar detection probabilities, a post-test simulation is performed to distinguish them and determine the possible single stuck-at fault that could produce the observed test results. Simulation is also needed to calculate the fault-free signature of the quotient stream compacted serially into the LFSR shown in Figure 3.5.

Another modulo-T binary counter is used to record the first occurrence of a "1" in the error stream by incrementing its value until the occurrence is detected. The use of this first failing-pattern information, combined with the additional LFSR signature, provides a fine-grain diagnostic resolution through limited post-test simulation [5].

Before stating the complete DAPPER method, some final points should be mentioned. The expected weight ($E(w_f)$) of the error sequence associated with a fault f with detection probability p_f is equal to:

$$E(w_f) = p_f T = \mu \ . \tag{3.11}$$

Since the applied test patterns are pseudorandom and independent in nature, and using the fact that successive outputs of a combinational circuit are independent when the inputs are independent [5], then the independent error model [121] can be employed. This model assumes that errors occur randomly and independently at a given circuit output with some fixed probability p_f equal to the random-pattern detection probability [5].

Under this error model, the errors are binomially distributed. Hence, the standard deviation (σ) is equal to:

$$\sigma = \sqrt{p_f(1-p_f)T} \ , \tag{3.12}$$

and by the Central Limit Theorem [122]:

$$P\left(|w_f - E(w_f)| \geq x\right) = N\left(\frac{|x-\mu| - 0.5}{\sigma}\right) \ , \tag{3.13}$$

where $N(z)$ is the probability of a standard normal random variable being at least z standard deviations from the mean. These probabilities (denoted the p_f-values) give the likelihood of a particular fault causing the observed output [5].

At this point the DAPPER method can be stated as follows:

1. The MINFSR, LFSR, Weight counter, and First-Fail counter are all cleared.

2. A sequence of T tests is applied by the TPG and the values of the LFSR and counters are observed at the end of the last test.

3. If the observed LFSR signature is identical to its pre-calculated reference and the weight counter is still in the all zero state, then no faults are detected and no diagnosis is required.

4. If the LFSR signature is different from its reference, then the MUT contains a fault. The diagnostic post-test simulation steps are as follows:

   (a) Select a fault which has not been simulated, whose $E(w_f)$ is closest to the observed one, and its p_f-value is sufficiently large to justify simulation.

   (b) Perform post-test simulation on the selected fault. The simulation can stop if the simulated fault fails on a pattern before the observed first failing-pattern as indicated by the "First-Fail" counter, or the simulated fault does not fail on the observed First-Fail pattern.

   (c) If the simulated fault in the previous step fails on the First-Fail test pattern, then simulation is performed on all T patterns. If both the simulation signature and Weight counter match the ones generated by the MUT, then the fault has been located.

   (d) If the fault is not located, then the diagnosis steps are repeated with another candidate.

The diagnostic *resolution* of this method is determined by the size of the fault equivalent classes resulting from diagnosis. Faults with low detection probabilities potentially have identical behavior that results in a reduced resolution (larger fault equivalence classes). The diagnostic resolution can be improved by increasing the test length and improving the accuracy of the estimated detection probabilities used in calculating the expected weights [5].

Under certain circumstances, this method may not even result in a unique diagnosis to a single fault equivalence class. This happens when a fault goes undetected with error cancellation in the MINFSR. This effect was not observed in any of the experiments in [5], and thus seems highly unlikely. It should be noted that if no error cancellation occurs in the MINFSR, then signature aliasing in the LFSR can be detected since the weight counter will have a non-zero value.

Some final comments are worth mentioning before leaving this method. First, the major deficiency of this method is the large memory required for storing the fault-free quotient stream. In addition, this technique only applies to strictly single stuck-at faults. This scheme could potentially require a considerable area overhead for the additional test and diagnosis circuitry.

Implementing the test and diagnosis hardware on-chip provides some kind of BISD capability. However, this need not be the case. The TPG, MINFSR, LFSR, counters, and quotient sequence generator can all be implemented on a separate test chip that can be shared by different MUTs. In this case, there is no hardware overhead in the chip containing the MUT itself.

Finally, this method has an advantage over the previous method with intermediate signatures, in that it does not require the collection of the actual circuit output values or the intermediate signatures. This in turn speeds up the overall diagnostic process.

3.2 Identifying the Failing Test Patterns

Identifying the failing tests in a signature analyzer is an alternative BIST diagnostic method. Once the failing tests are identified, then classical diagnostic methods such as post-test simulation can be used with these tests to locate the actual faults in the MUT. This is simpler and faster than the alternative of logging out the response of each test pattern and comparing it against a simulation [72].

Identifying the failing test patterns in turn requires the identification of the error bits (failing bits) in the output response sequence of the MUT, for each of the applied test patterns.

Four methods are presented in the following sections to identify the failing tests and their corresponding errors in the TRS. The test and diagnosis models considered with these methods are still the ones shown in Figure 3.2.

The diagnostic techniques begin with the error signature, $S_e \neq 0$, and *try* to construct an error sequence E that fully explains S_e. Given such an error sequence, then identifying the failing test patterns is trivial.

Because these methods operate solely upon the limited information contained in the compacted faulty signature, they are subject to *diagnostic aliasing*. Diagnostic aliasing

occurs when multiple tests in the BIST session fail and the method indicates fewer test failures, or when *false alarms* are produced [72].

3.2.1 Identifying Failing Tests With Reciprocal SARs

The first approach to determining failing test patterns from the faulty signature was developed in [72]. The SAR used for testing was implemented as a Type-2 LFSR with a primitive characteristic polynomial $f(x)$ of degree m.

The method begins with the error signature S_e and tries to identify an error sequence E that fully explains S_e. It subscribes to the principle of economy in the explanation. Accordingly, S_e is explained with a single failing test first. If it cannot, two failing tests are tried, and so on.

Following the conventions established earlier in Chapter 2, the t^{th} failing pattern corresponds to the t^{th} bit in E. Suppose first that only a single test $t = T - i - 1$ $(0 \leq i < T)$ has failed. Then $E(x) = x^i$. The required diagnostic action is simply to find i. Since

$$S_e = x^i \bmod G(x) \tag{3.14}$$

then,

$$x^i = S_e \bmod G^{-1}(x) \ , \tag{3.15}$$

where $G^{-1}(x) = f(x)$.

In reference to the above relation, the simplest procedure for computing i is to implement a reciprocal LFSR with divisor polynomial $f(x)$, seed it with the error signature S_e, and run it forward. On each cycle, the reciprocal register decrements the value of i in x^i by one. After the first cycle, the register contains $x^{i-1} \bmod f(x)$. After the second cycle, it contains $x^{i-2} \bmod f(x)$. And after i cycles, it contains $x^0 \bmod f(x)$ which is recognized as the m-bit pattern "$100\cdots000$".

A modulo-T binary counter can be used to count the number of cycles required to encounter the m-bit target pattern. The reciprocal LFSR and counter can be implemented in either hardware or software. Clearly, the amount of computation required to determine the failing test depends strictly on the error location in E.

To summarize the procedure for identifying a single failing test t_i out of the T applied tests:

1. Compute the error signature S_e. If $S_e = 0$, then no errors are detected and diagnosis is not required. Else, there exists at least one failing test.

2. Seed the reciprocal register with S_e and clear the modulo-T binary counter.

3. Cycle the reciprocal register and increment the counter until the m-bit pattern "$100\cdots00$" is encountered.

4. The content of the counter is $T - t - 1$ and the diagnosis is complete.

The method is applied to a MISR in exactly the same manner as described above, except for the uncertainty as to which MISR input carried the erroneous bit. A window of uncertainty equal to the number of MISR inputs is unavoidable. Thus, the method can only bound the failing test between test t_i and test t_{i+m-1} inclusively.

If the target pattern "$100\cdots00$" is not encountered in T cycles, then there exist at least two failing tests. Diagnosing two failing tests is somewhat more complex and computationally intensive than one failing test. This is increasingly true for more failing tests [72]. Hence from a practical point of view, the method has very limited use, for if a MUT is faulty, then it will most likely produce many erroneous bits, not just one or two.

In order for the method to work, the degree of the input polynomial to the SAR must be limited to the order of $G(x)$, i.e., $T < 2^m - 1$. As mentioned previously in Chapter 2, if this condition is not met, then signatures begin to repeat. Thus, there is a possibility that the pattern "$100\cdots00$" will be encountered on several cycles, which can cause misdiagnosis. Therefore, the SAR's length (and hence the length of its cycle) must be matched to the test length to meet this restriction. This in turn results in large SARs being used.

Finally, it should be mentioned that when a MUT produces multiple erroneous bits, the method may alias to fewer tests or even to a single failing test [72]. Assuming that the faulty signatures are evenly distributed over the signature space, then the probability of incorrectly identifying multiple test failures as a single failing test is bounded by [72]:

$$P_{DA} \leq \frac{T}{2^m}\ , \quad T > m\ . \tag{3.16}$$

Clearly, the probability of diagnostic aliasing approaches unity as the number of applied tests approaches the order of the SAR's characteristic polynomial. If $T \leq m$, then there is no aliasing since the entire error sequence is in the SAR.

3.2.2 Identifying Failing Tests With Cycling SARs

Another approach to identifying the failing test patterns was proposed in [111]. The method is based on measuring signatures with a number of cycling LFSRs. Again, the method begins with the error signature S_e and tries to identify an error sequence E that fully explains it.

Assume for the moment that there is only a single failing test. Then $E(x) = x^i$, corresponding to the $(T - i - 1)^{\text{th}}$ failing test pattern. The required diagnostic action is to simply find i.

As mentioned in Chapter 2, a captured erroneous bit in a cycling LFSR merely cycles around the register in step with the shifting clock, in the absence of aliasing. A cycling LFSR also has the property that each bit in the register is independent of the others [111]. With these properties, it is apparent that S_e will also have a single "1" (called a *foot print*) in a field of $m - 1$ 0s. Thus,

$$S_e(x) = x^p \ , \tag{3.17}$$

where p is the position of the foot print. Clearly,

$$\begin{aligned} x^p &= E(x) \bmod (1 + x^m) && (3.18) \\ &= x^i \bmod (1 + x^m) \ , && (3.19) \end{aligned}$$

or equivalently,

$$p = i \bmod m \ . \tag{3.20}$$

Reversing the discussion, if S_e has a single foot print at position p $(0 \leq p < m)$, then the error sequence has an odd number of 1s in some or all the positions given by:

$$i = p + \sigma m \ , \tag{3.21}$$

where $(\sigma \geq 0)$ is a positive integer. A single foot print in S_e identifies a set of possible failing tests which are distance m apart. Thus, there is an uncertainty with regard to the actual number of errors that have occurred and their positions in the error sequence.

As with the reciprocal SAR method, the uncertainty can be avoided if the test length (T) is made less than the cycle length of the SAR (m for an m-bit cycling register). Obviously, the diagnostic resolution and the maximum test length associated with a single

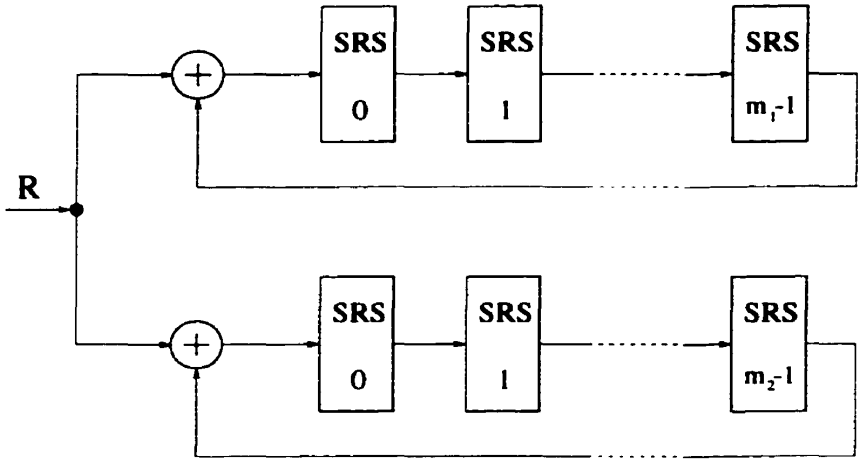


Figure 3.6: BIST compaction with multiple cycling registers.

cycling register are not satisfactory. This difficulty was overcome in [111] by using two cycling registers to compact the output response in parallel. The registers shown in Figure 3.6 must have relatively prime lengths, m_1 and m_2.

Assuming that the foot prints in the measured signatures are at positions p_1 and p_2 for the first and second cycling LFSRs, respectively, then i can be easily deduced by solving (in software) the two linear equations :

$$i = p_1 + \sigma_1 m_1 \ , \tag{3.22}$$

$$i = p_2 + \sigma_2 m_2 \ . \tag{3.23}$$

It has been proven in [111] that if the test length is limited to the least common multiple of the orders of the two cycling registers, $\text{lcm}(m_1, m_2) = m_1 m_2$, then all single errors can be uniquely identified from the measured signatures in both cycling registers. For large test sets, the size of the cycling registers could become rather large as well.

Identifying multiple errors follows exactly as the single error case. Let $P_1 = \{p_{1,1}, p_{1,2}, \cdots, p_{1,i}\}$ be the positions of the foot prints in S_{e_1}, and $P_2 = \{p_{2,1}, p_{2,2}, \cdots, p_{2,j}\}$ be the positions of the foot prints in S_{e_2}. Then the locations of the erroneous bits in E can be found by solving Equations 3.22 and 3.23 for all (i, j) pairs of P_1 and P_2.

Thus, the complexity of diagnosing multiple failing tests is the same as that of diagnosing a single failing test, except that it is possible to get diagnostic aliasing [111]. As a result, it is possible for the diagnostic procedure to declare error-carrying positions as

error-free (*type-1 aliasing*), and/or to declare error-free positions as error-carriers (*type-2 aliasing*).

Clearly, a type-1 aliasing is more severe than a type-2 since it provides no trace of the hidden errors, while a type-2 aliasing results only in *false alarms* [111]. Re-testing the MUT with the identified failing test patterns will reveal all false alarms which can then be discarded.

As mentioned above, there can be no aliasing if the response sequence has a single erroneous bit. Furthermore, double failing tests may result in only a type-2 aliasing. The reason is that for a type-1 aliasing to occur, at least one of the cycling registers must lose one of its foot prints. But since foot prints are lost in pairs (which are a multiple of m_1, or a multiple of m_2 apart), one of the registers will show no foot prints, while the other will show both. Consequently, only a type-2 aliasing may occur. Hence, a type-1 aliasing can only occur when the number of failing tests is greater than or equal to three.

Diagnostic aliasing was reported to increase with the number of errors in the MUT's output stream [111]. Conversely, the diagnostic aliasing probability decreases as the number of stages in the smallest cycling register increases. The increase in aliasing probability is generally caused by foot prints being cancelled in the cycling registers. Aliasing becomes definite when the number of errors exceeds the number of stages in the smallest cycling register. The reason for the latter is that a cycling register cannot retain information on a number of errors greater than its size.

In order to allow for a practical test length and reduced diagnostic aliasing, multiple cycling registers of relatively prime lengths can be used at the expense of additional hardware. For example, three SARs can be constructed to identify any single or double errors, and extend the test length to $\text{lcm}(m_1, m_2, m_3) = m_1 m_2 m_3$ [111]. The failing tests are now computed by solving a set of three linear equations.

In conclusion, there is a trade off between the allowed test length and the quality of diagnosis on the one hand, and the hardware overhead cost on the other hand. If the number of errors entering the cycling registers is small compared to their sizes (say no more than half of the shortest register), this diagnostic method will identify most if not all failing test patterns very efficiently. This in turn has a quite noticeable impact on simplifying the diagnostic process in BIST circuits. The efficiency of the method decreases as the number of failing tests increases.

Compared to the reciprocal-SAR method [72], the current method is more *practical*

(not to the point where it can actually be used in real VLSI circuits), but it requires more hardware overhead if good diagnostic resolution is to be obtained.

3.2.3 Identifying Failing Tests With Conventional SARs

In [64, 110, 123, 124, 125, 126], another diagnostic approach is presented with the objective of identifying the location(s) of the erroneous bit(s) in the test response sequence and their corresponding failing tests.

The basic idea is to analyze the error signature S_e to *estimate* and *guide* the search for potential error locations in E, starting with the assumption of a small number of errors. Once the possible error locations are determined, matching tests are performed to confirm them. In a matching test, the actual outputs of the MUT are compared to the fault-free ones for the predicted *likely-failing* test. In exhaustive testing, the number of matching tests is always equal to the test length.

In essence, the identification of a failing test requires the determination of the relation between an error location and its syndrome for a given SAR [64]. Consider first a SAR implemented as a Type-2 LFSR [110, 123, 124, 125]. Following the conventions formulated in Chapter 2, if there are exactly l tests in error, then $E(x)$ can be written as the sum of l monomials:

$$E(x) = x^{i_1} + x^{i_2} + \cdots + x^{i_l} \quad , \tag{3.24}$$

where $i_j \neq i_k \in \{0, 1, \ldots, T-1\}$, and T is the length of the test sequence.

The error signature can be expanded as the sum of l singleton signatures:

$$S_e = S_{i_1} \oplus S_{i_2} \oplus \cdots \oplus S_{i_l} \quad , \tag{3.25}$$

where S_{i_j} is the signature corresponding to the single error bit polynomial $E(x) = x^{i_j}$.

It has been proven in [91] that singleton signatures are *distinct* provided that the degree of the error polynomial, and thus the test length, is less than the order (d) of the divisor polynomial $G(x)$ implementing the LFSR. In order to maximize d, $G(x)$ is assumed to be primitive. If $G(x)$ has degree m, then d is equal to its vector space of $(2^m - 1)$.

Detecting and locating any set of l errors requires the condition that any set of l or less errors are not masked by any other set of l or less errors. In that case, all possible sums of at most l distinct singleton signatures are distinct modulo $G(x)$ [91].

Combining the above conditions, then all possible sums of at most $2l$ distinct singleton signatures must be linearly independent so that:

$$\left(\sum_{j=0}^{2l-1} a_j x^{i_j}\right) \bmod G(x) \neq 0 \ , \tag{3.26}$$

where $a_j \in \{0,1\}$, $i_j \in \{0,1,\ldots,T-1\}$, and $T \leq d$ [127].

This is equivalent to the Coding Theory requirement that the minimum distance (d) between any two codes should be $2l+1$ for a code designed to uniquely detect and correct at most l errors [84]. From a practical point of view, this constraint imposes stringent requirements on the divisor polynomial and the test length [64]. As the number of errors increases, the degree of $G(x)$ grows rapidly, approaching the test length.

The above constraint cannot be met in most applications of signature analysis. As a result, the procedure for identifying the erroneous tests requires a process of interpreting the error signature in terms of the single failing-test signatures or syndromes [64]. To this end, the most straightforward approach is to construct a Look-Up Table (LUT) which gives the erroneous tests as a function of the singleton syndromes.

Given the $G(x)$ used in compacting the TRS and the length of the test T, these S_i signatures can be easily computed independent of the response sequence. The computation can be done in software before the diagnostic routine is invoked.

When a faulty signature is observed, the search for the candidate erroneous tests proceeds as follows:

1. Interpret S_e as being due to a single failing test with $E(x) = x^i$, $0 \leq i < T$. Identify an error location i such that $S_e = S_i$. If there exists an $S_i = S_e$, apply a matching test at location i. If the test confirms the error for that location, the error and the corresponding failing test are located. In this case, the number of matching tests is reduced from T, as required by exhaustive testing, to just one. If the matching test fails or if no $S_i = S_e$ is found for all i, the assumption of a single cycle error fails.Thus, S_e must be caused by multiple failing tests. All matching signatures ($S_i = S_e$) are bitwise XOR'ed with S_e in order to remove them from its expansion before proceeding to Step 2.

2. Interpret S_e as being due to multiple failing tests. Starting with double errors ($l = 2$), expand S_e in terms of two singleton signatures. For each combination, a singleton

signature S_i is XOR'ed with another one S_j, and the result is compared with S_e. If a match is found then a matching test is performed to verify the corresponding locations. If the test confirms both error locations, the errors and their corresponding failing tests are located. If the matching test fails or if S_e cannot be expanded by two singleton signatures, assume triple errors ($l = 3$) and so on. The search in each step continues after removing the singleton signatures of the matching bit locations as explained in Step 1.

In general, the number of error combinations depends on the divisor polynomial, the test length, and the number of failing tests. After removing j syndromes from the expansion of S_e, the number of different possible error sequences that contain exactly l non-zero entries is equal to

$$\binom{T-j}{l} . \tag{3.27}$$

Unfortunately, the results of this technique are often misleading. When the error signature is confirmed by i errors in the response sequence, it is possible that the i single test errors are part of a larger multiple error sequence [110]. The remaining uncovered errors in the sequence become *escapes* to the divisor polynomial of the SAR. For example, if $i = 1$, then the minimum number of such error escapes is three [110]. This is because the singleton signatures have a minimum code distance of one when $G(x)$ is primitive and $T < 2^m - 1$.

It is well known that for a divisor polynomial of degree m and a test sequence of length T, the number of aliasing error sequences is equal to $2^{T-m} - 1$, where one sequence corresponds to the all zeros (error free) one [110]. When the error signature is confirmed by i errors, the degree of the error polynomial is reduced by i. Hence, the probability of error escapes [110, 124] or *diagnostic aliasing* is bounded by:

$$P_{DA} \leq \frac{2^{T-m-i} - 1}{2^T} \tag{3.28}$$

$$\approx \frac{1}{2^{m+i}} . \tag{3.29}$$

This diagnostic approach can also be applied to SARs implementing a Type-1 or Type-2 MISRs [64, 126]. From a practical point of view, when a MISR is used, one may not be only interested in locating the erroneous tests, but also in identifying the MISR input(s)

which carried the error(s). Assume an m-stage MISR with a primitive divisor polynomial, and a test length less than $2^m - 1$.

Since single test errors occurring on different inputs of a MISR have unique syndromes [64], a single test error will always be detected. Consequently, if the test at which errors occur is known, S_e will pinpoint the error capturing MISR inputs . For a single test error(s), this relation is characterized by Equation 2.43:

$$S_{e,t} = E_t C^{T-1-t} \quad . \tag{3.30}$$

Thus when the erroneous test location "t" is known, the erroneous MISR input(s) can be identified by solving the above equation for E_t. The computation can be done in a complex field or in GF(2^m). When evaluating over a complex field, there is a potential problem of round-off errors in the computation. This cannot be neglected when "$T - 1 - t$" is large [64].

Conversely, if the faulty inputs are known, then a single failing test can be uniquely identified [126]. This is a generalization for the LFSR case. Thus, if the compacted errors do not vary among the MISR inputs, then the probability of diagnostic aliasing for a MISR would be the same as that of an LFSR. Identifying the erroneous tests requires solving Equation 3.30 for "t".

Finally, when multiple errors with different patterns occur in multiple tests, the final error signature is described by Equation 2.44. The interpretation of S_e would then involve the identification of the tests and the MISR inputs where errors occurred.

In summary, this simple solution for identifying multiple failing tests can identify more failing tests than both the reciprocal SAR method [72] and the cycling SAR method [111]. The method is based on the observation that the signature of a multiple error sequence can be computed from the signatures of single error sequences. The search space of the error location(s) is targeted by expanding the error signature in terms of the singleton signatures. The candidate error locations are then confirmed by matching tests.

In general, the practicality of such an approach depends on the efficiency of computing error syndromes, the efficient generation of the valid error combinations, and the test length relative to the order of the divisor polynomial implemented by the SAR [64]. When $T \ll 2^m - 1$, the search for failing tests can be effectively guided by the expansion of the error signature. Furthermore, when a matching test is difficult, the method provides an educated guess of the error(s) that may have occurred [110].

The diagnostic algorithm can be readily implemented in software with the faulty signature as its only input. It always leads to error identification in a *homing-in* manner. Thus, when the multiplicity of failing tests is small, the error location(s) can be determined rapidly. The number of matching tests can be significantly reduced with a probability of diagnostic aliasing. Therefore, the method has a great time saving advantage over exhaustive test matching.

As the number of failing tests increases, the performance of this "search before test" algorithm degrades to that of exhaustive test matching [125].

One disadvantage of this method is that the number of entries in the LUT may become excessive for long test sequences. Another disadvantage is that it requires the storage of all MUT responses for the purpose of test matching. This may not be practical when the test length is large, and it negates the benefits of BIST compaction.

3.2.4 Identifying Failing Tests With Composite SARs

In [90], an efficient technique for single error bit identification is presented in conjunction with a SAR having a divisor polynomial $G(x) = G_1(x)G_2(x)$, where $G_1(x)$ and $G_2(x)$ are polynomials of different degrees.

The method makes use of the fact that if $G_1(x), \ldots, G_l(x)$ are pairwise *relatively-prime* nonzero polynomials and $G(x) = G_1(x) \times \cdots \times G_l(x)$, then,

$$\text{ord}\,(G(x)) = \text{lcm}\,(\text{ord}\,(G_1(x)), \ldots, \text{ord}\,(G_l(x))) \ , \tag{3.31}$$

where ord and lcm denote the order and least common multiple, respectively [128].

Therefore, by using two SARs with divisor polynomials $G_1(x) = G_m(x)$ and $G_2(x) = G_{m+1}(x)$, the degree of the error polynomial to the SAR can be extended to

$$d \leq \text{lcm}\,(\text{ord}\,(G_1(x)), \text{ord}\,(G_2(x))) \ . \tag{3.32}$$

As the degree of the error polynomial exceeds d, the signatures start to repeat as mentioned in Section 2.5.5.

Although any pair of relatively-prime, primitive or non-primitive, polynomials can be used, the class of polynomials given by $G_m(x) = x^m + x^{m-1} + 1$ was specifically investigated in [90].

In a BIST application, the SAR can be constructed as a single SAR with divisor polynomial $G(x) = G_1(x)G_2(x)$, or as two separate SARs with divisor polynomials $G_1(x)$ and $G_2(x)$. In either implementation, it is crucial that the SARs be constructed as a Type-2 LFSR since a Type-1 SAR will not always give the correct signature [1].

Both implementations are equivalent in terms of the number of XOR gates and SRSs required to implement the SAR polynomials. However, additional signature processing is required, prior to identifying the error bit, when the single SAR implementation is used [90]. Consequently, the implementation of separate SARs is considered in [90].

This diagnostic approach is efficient in terms of the area overhead when compared to the cycling SARs technique described earlier in Section 3.2.2. Yet this approach extends the degree of the error polynomial to the least common multiple of the *orders* of the two divisor polynomials, as opposed to the least common multiple of the *degrees* of the two divisor polynomials in the cycling SAR technique.

Following the conventions established earlier in Chapter 2, suppose that only a single test $t = T - i - 1$ $(0 \leq i < T)$ has failed with $E(x) = x^i$. The error signatures, $\{S_{e_1}(x), S_{e_2}(x)\}$, formed in the two SARs are then equal to:

$$S_{e_1}(x) = x^i \bmod G_1(x) \ , \tag{3.33}$$

$$S_{e_2}(x) = x^i \bmod G_2(x) \ . \tag{3.34}$$

Unique combinations of the two signatures are obtained for single bit error polynomials of degree up to lcm(ord($G_1(x)$),ord($G_2(x)$)). Therefore, all single errors can be detected and identified.

A LUT is used to give the pair of signatures as a function of each erroneous bit in the test response sequence. The number of entries (L) required for such a LUT is equal to:

$$L = \text{ord}\,(G_1(x)) + \text{ord}\,(G_2(x)) \ . \tag{3.35}$$

The maximum L, encountered when both polynomials are primitive, is given by:

$$L_{\max} = 2^m + 2^{m+1} \tag{3.36}$$

$$= 3 \cdot 2^m \ . \tag{3.37}$$

Since the maximum number of LUT entries is sufficiently small, the use of LUTs in the error identification becomes a viable and practical approach when compared to a SAR constructed from a single primitive divisor polynomial [90].

Let i_1 and i_2 represent the candidate error bits obtained from the LUT in conjunction with the signatures $S_{e_1}(x)$ and $S_{e_2}(x)$, respectively. The single error bit identification algorithm proceeds as follows [90]:

1. If $S_{e_1}(x) = 0$ or $S_{e_2}(x) = 0$, but not both, then there are multiple errors in the test response sequence and the algorithm terminates since it is designed only for single bit errors.

2. If $S_{e_1}(x) \neq 0$ and $S_{e_2}(x) \neq 0$, the candidate error bits are obtained from the LUT for each SAR. If either error signature does not exist in its respective LUT, then there are multiple errors in the TRS and the algorithm terminates. If both error signatures are valid, the erroneous bit "i" is identified, satisfying the following relations:

$$i_1 = i \bmod \text{ord}(G_1(x)) \ , \tag{3.38}$$

$$i_2 = i \bmod \text{ord}(G_2(x)) \ . \tag{3.39}$$

   The error identification algorithm terminates successfully at this point.

The probability of diagnostic aliasing has also been investigated in [90] to quantify the likelihood of the algorithm identifying a single erroneous bit when in fact multiple errors exist in the TRS. Based on simulation results, it was concluded that a reasonable estimate can be given by [90]:

$$P_{DA} = \frac{d}{(2^{2m+1})} \ . \tag{3.40}$$

It should be noted that P_{DA} is the ratio of the compactor's order, to the order of a *primitive* polynomial of degree $2m + 1$. Furthermore, the experimental data in [90] has shown that P_{DA} is a function of the divisor polynomials used to implement the SARs. Specifically, it has been shown that the probability of diagnostic aliasing can be significantly reduced using *non-primitive* divisor polynomials.

Hence, non-primitive polynomials of small degree can be used, such that the number of LUT entries is also small, without increasing P_{DA}. One concern in doing so is the potential increase in the probability of traditional signature aliasing (P_A). Yet, P_A was also found to be lower with this approach, when compared to traditional SARs, due to the ability to detect more *combinations* of error bits. In fact, the probability of traditional signature aliasing has been found to be approximately that of a SAR constructed from a single primitive polynomial of degree $2m + 1$ [90]. Hence, $P_A \approx 2^{-(2m+1)}$.

Single and Double Error Bits Detection and Identification

The previous technique has been extended in [127] to include both single and double bit error identification with SARs implemented as LFSRs or MISRs. Such a scheme employs similar techniques to those used in BCH codes [84] for double error detection and correction.

Consider first the case of LFSRs. It has been proven in [127] that all single and double bit errors can be detected and identified as long as the divisor polynomial for the SAR is given by $G(x) = G_1(x)G_1^{-1}(x)$, where $G_1^{-1}(x)$ is the reciprocal of $G_1(x)$, a *primitive* polynomial of *odd* degree.

It should be noted that $G(x)$ is not a primitive polynomial. Also, like the single error bit technique, the error polynomial is limited to the order of the divisor polynomial $G(x)$. Since the orders of $G_1(x)$ and $G_1^{-1}(x)$ are equal, then $\text{ord}(G(x)) = \text{ord}(G_1(x)) = d$.

Suppose that tests $t_1 = T - j_1 - 1$ and $t_2 = T - j_2 - 1$, $0 \leq j_1, j_2 < T$, have failed. The error polynomial is then equal to $E(x) = x^{j_1} + x^{j_2}$. The required diagnostic action is simply to find j_1 and j_2.

The error signatures $(S_{e_1}(x), S_{e_2}(x))$ formed in the two SARs are equal to:

$$\begin{aligned} S_{e_1}(x) &= (x^{j_1} + x^{j_2}) \bmod G_1(x) && (3.41)\\ &= x^{i_1} \bmod G_1(x)\ , && (3.42)\\ S_{e_2}(x) &= (x^{j_1} + x^{j_2}) \bmod G_1^{-1}(x) && (3.43)\\ &= x^{i_2} \bmod G_1^{-1}(x)\ . && (3.44) \end{aligned}$$

The candidate error bits (i_1, i_2) are obtained from a LUT based on the observed $S_{e_1}(x)$ and $S_{e_2}(x)$, respectively.

The errors identification algorithm is implemented in system diagnostic software. It essentially follows in the same manner as in the single error bit case with the following modifications:

1. The case where $i_1 = i_2 = i$ indicates that a single error bit has been identified at the i^{th} position of the test response sequence.

2. The case where $i_1 \neq i_2$, indicates that there is more than one erroneous bit. The basic principle is to find an error polynomial which satisfies the relationships of Equations 3.41 and 3.43. Let $h = (i_1 + i_2) \bmod d$. Set $j_1 = 0$ and $j_2 = h$, and check

to see if the relationships are satisfied. Increment and decrement (modulo-d) the values of j_1 and j_2, respectively, until the two relationships are satisfied. If such pair of j_1 and j_2 is found, the algorithm terminates successfully. Otherwise, there are more than two erroneous bits in the TRS and the algorithm terminates unsuccessfully.

The algorithm described above can also be applied to SARs implemented as MISRs. In this case, the error identification algorithm must determine the test(s) in which the error(s) occurred and the MISR input(s) that captured the erroneous bit(s).

In order for the above algorithm to be directly applicable, the multiple input stream to the MISR is converted to an equivalent single input sequence entering an LFSR as described in Section 2.5.5. The number of test patterns T applied to an m-bit MISR must satisfy the relation $T < d - m - 1$, where d is the order of $G_1(x)$ used to construct the MISR.

However in this case , the diagnostic algorithm faces the additional possibility of double error-cancellation between the multiple inputs to the MISR. This possibility was overcome in [127] by running the test sequence three times while reordering the inputs to the MISR. During each execution of the BIST session, a 3-to-1 multiplexer is used at each MISR input to select between the normal ordering of inputs, reverse ordering of inputs, and the odd numbered inputs followed by the even ones. The identified errors in each execution are compared and a *majority* vote determines the final diagnostic results. This allows for error cancellation in one of the error sequences.

The diagnostic aliasing probability with this approach was found to average about 50% and to be relatively independent of the number of stages in the MISR [127].

In cases where the number of errors is greater than four, twice the maximum number of errors the algorithm was designed for, a small number of occurrences of traditional signature aliasing have been found [127]. The aliasing magnitude was about the same as that of a single MISR implemented with a primitive polynomial of degree twice that of $G_1(x)$ [127]. No such aliasing is encountered when the number of error bits is less than or equal to four, which verifies the fact that all double errors are detected.

Multiple Error Bits Detection and Identification

Finally, the error bit identification algorithm has been extended in [91] to detect and identify any specified number of errors in the test response sequence.

To detect and identify l errors in a T-bit response sequence, the SAR must be constructed with the divisor polynomial:

$$G(x) = \text{lcm}(G_1(x), G_3(x), \ldots, G_{2l-1}(x)) \ , \tag{3.45}$$

where lcm represents the least common multiple of the polynomials:

$$G_i(x) = \text{Res}_t(G_1(t), x - t^i), \quad i = 3, \ldots, 2l-1 \ . \tag{3.46}$$

Res_t denotes the t-Resultant [128] and $G_1(x)$ is any polynomial (not necessarily primitive) whose order is greater than T.

The $G_i(x)$ polynomials are similar to the generating polynomials used with the construction of BCH cyclic codes [84]. An l-error correcting BCH code can be used to detect $2l$ bit-errors and locate l errors in any sequence without aliasing.

Unlike the single and double error identification methods, the SAR here is a conventional one with $G(x)$ as its divisor polynomial. The size of the SAR is determined by the test length and the number of erroneous bits to be identified. For example, a 10-bit, 15-bit, and 20-bit SAR were used in [91] to respectively identify 2, 3, and 4 errors in a test session of length 30.

In general, the number of SRSs required to implement the SAR is given by the product of l and the degree of $G(x)$ [91]. This number represents an upper bound and can be less in some cases as a result of the lcm construction of $G(x)$. For example, the identification of three bits in an error polynomial of degree 4,095 would only require a 36-bit SAR [91].

Due to the hardware overhead, l is kept to 4 or less. Some solutions have been suggested in [91] to overcome prohibitively large SARs being required.

Given the faulty signature, the error bit identification can be done in software with the algorithm developed in [91]. Due to the lengthy mathematical details of the algorithm and space limitations, the algorithm will not be presented here. It should be mentioned, however, that for sequences with more than 4 erroneous bits, the diagnostic aliasing ranges from 48% to 65%.

Chapter 4

Board, MCM and System Level Diagnosis

A VLSI system, board, or MCM is usually composed of n Replacable Units (RUs) interconnected together. At the system level, the RU can be a PCB, MCM or a VLSI processor. At the board or MCM level, the RU is usually a chip.

It is costly to isolate hardware failures to an RU using traditional software diagnostic procedures based on exercising the normal function of the module [23]. The rapid reduction of hardware cost suggests that modules should be designed to ease the diagnosis process and ensure reliability at a reasonable cost. Hence the concept of Design For Diagnosability (DFD). *Diagnosability* is defined as the ability to detect errors and locate faults to an RU, easily and cost-effectively [23]. The diagnosability of a VLSI module has a great impact on the cost of testing and diagnosis.

In a diagnosable VLSI system, board or MCM, fault diagnosis is usually realized by additional hardware instead of traditional software procedures, so that the computational space and time for fault isolation are reduced substantially [23, 47]. This in turn reduces the delay in isolating and repairing the faulty unit(s) either on-site or at the field service center.

The accurate and automatic diagnosis obtained with DFD techniques minimizes the module down time, which translates into saving dollars and results in highly reliable digital modules.

A lot of research has been devoted to the problem of diagnosis in VLSI systems, PCBs,

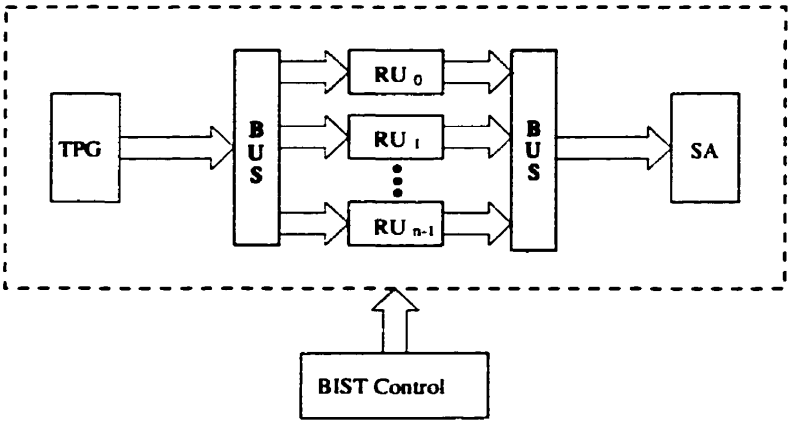


Figure 4.1: General form of a centralized BIST architecture.

and MCMs. Different diagnostic BIST architectures have been developed to be used with time compaction using signature analysis, space compaction, or their combination. In the remainder of this chapter, a concise description of these methods will be provided, emphasizing their applicability, practicality, pros, cons, and cost.

4.1 BIST Fault Diagnosis With Time Compaction

Various system, board, and MCM self-test and diagnosis architectures have been suggested in the literature. They are based on signature analysis. There is no one *best* architecture, but rather a collection of possibilities, the choice of which depends upon the application, diagnostic performance, hardware overhead, and test time.

All of these architectures fall in three main categories. The differences between them lie in the test bus used (the boundary scan TAP bus or system bus), and the TPG and ORA structures associated with them. However, the basic testing and diagnosis procedure is the same across all architectures. This section summarizes the architectures which have been suggested or used in the past.

4.1.1 Centralized Diagnostic BIST Architectures

The general form of a centralized BIST architecture is shown in Figure 4.1.

In this architecture, several RUs on the MUT share the same TPG and SA circuitry.

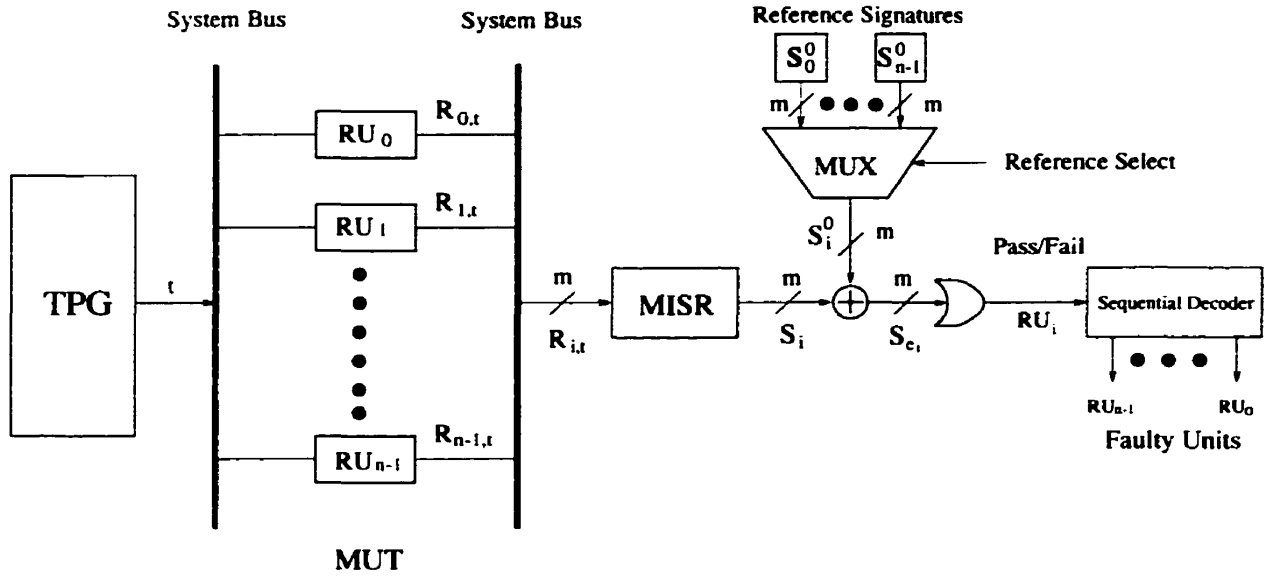


Figure 4.2: Centralized serial diagnosis using the system bus [6].

This leads to a reduced hardware overhead, but increases test time [3]. During testing, the BIST controller communicates with the RUs using system or test buses. It steps the RUs through the BIST test sequence as explained next.

Diagnosis Using the System Bus

The diagnostic architecture for the straightforward serial approach is shown in Figure 4.2 [6, 129].

In this architecture, each unit (processor, board, MCM or chip) on the original module is tested separately. An LFSR is used as the on-module test pattern generator. The diagnostic procedure consists of generating a new pseudorandom test pattern t ($0 \leq t < T$), transferring it via the system bus to the i^{th} unit being tested, and transferring its response $R_{i,t} \in \text{GF}(2^m)$ to an m-bit MISR used as a time compactor. After applying all T tests, the compacted signature S_i is compared to its pre-calculated error-free reference S_i^0 to obtain the error signature $S_{e_i} = S_i \oplus S_i^0$. If $S_{e_i} \neq 0$, then a *fail* bit is transferred to the diagnostic decoder. Otherwise, a *pass* bit is transferred to it.

The above diagnostic procedure is repeated serially for each of the n RUs on the MUT. Once the last unit is tested, the n-output sequential decoder flags the faulty RU(s) based on the transferred fail/pass information.

The hardware overhead of this centralized architecture consists of the m-bit MISR, an m-bit comparator, an m-bit $n \times 1$ MUX, n m-bit reference signature registers, and a sequential decoder implemented as a simple FSM. Thus, the hardware complexity is of order $O(m)$. The time complexity is clearly proportional to $(n+1)T$, the number of units on the MUT and the number of applied pseudorandom tests.

It should be noted that the signature analyzer (MISR), can be replaced by a simple multiple-input adder to perform the time compaction [23]. This is especially suited to multiprocessor boards or systems where compaction can be done with a processor ALU [130, 131]. Generally speaking, there are many choices for the TPG and time compactor, and further research is needed to develop more efficient, centralized diagnostic designs.

Diagnosis Using the Boundary Scan TAP Bus

The boundary scan architecture enables a more systematic approach to testing and diagnosing modules [2, 132]. A full implementation of BIST and BS at the system, board or MCM level can greatly improve the accuracy of diagnosing such complex, high-density modules [133].

In these modules, the TAP bus serves as the common medium through which an RU is polled or scanned for diagnosis. The same RU signature can be used at all levels of packaging, which in turn simplifies the diagnosis process considerably. The RU reference signature needs to be calculated only once, and then can be migrated from the unit level to the higher module-level.

Various flavors of the architecture can be obtained depending on the structures of the TPG, signature analyzer, and the BS chain configurations [132].

In the first architectural group, all of the units on the module are organized logically in a single scan chain of n nodes, each representing a single unit [132, 134]. The *serial* TPG is implemented as an LFSR. The single-input signature analyzer is also implemented as an LFSR. This is shown in Figure 4.3.

During testing, all I/O pads of the individual units are connected in a single scan chain as shown in Figure 4.3(a). A unit diagnosis is performed by bypassing any number of units as shown in Figure 4.3(b), such that each unit is isolated from the remaining ones in the MUT.

Assuming multiple faulty units, locating the defective units is performed by BISTing

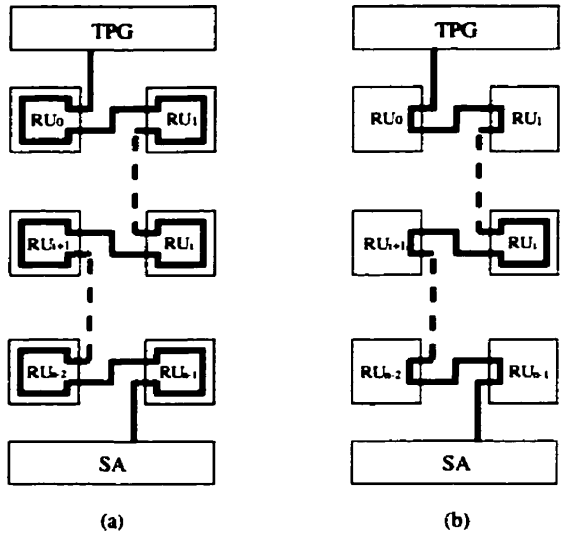


Figure 4.3: Centralized single BS chain architecture.

each one separately. This is accomplished by shifting a deterministic control sequence via the scan path into their IRs. According to the shifted instruction code, the *internal test* mode is activated for the unit under test using its BSR, while the *bypass* mode is activated for the remaining units, thus selecting their 1-bit BRs.

The pseudorandom test patterns are applied from the TPG to the TDI of the first unit in the chain, and are bypassed by all units preceding the TDI of the UUT. The output responses from the UUT, which exist in its BSR, are shifted out through its TDO, and bypassed through the units that succeed it. The TDO of the last unit in the chain is compacted by the LFSR. At the end of the BIST session, the final signature is compared with its corresponding pre-computed reference signature. If the two signatures are different, then a faulty unit is located.

This process is repeated for all units on the MUT. Hence, the number of BIST sessions required to achieve 100% diagnostic resolution is equal to the number of units (n) on the module [132]. The hardware overhead, however, is simpler than the one required when the system bus is employed. This is negated by the extra hardware added to each unit to implement the boundary scan feature. Note that if BIST is not implemented on the module, then a simple tester can be used instead [2].

In the other main architectural group, the n units on the MUT are organized logically as n BS chains as shown in Figure 4.4. This organization is generally called STUMPS (Self-Testing Using MISR and Parallel SRSG) [1, 47, 118].

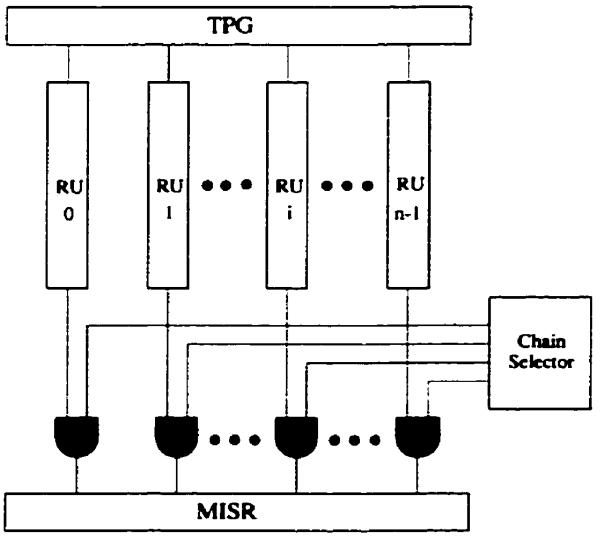


Figure 4.4: Centralized multiple BS chains architecture.

The pseudorandom patterns are generated by an LFSR on the module and are applied through the BS chains. The test results collected in the chains are scanned out into a single MISR where they are compacted into a final signature.

The chain selection controller consists of a counter, with each bit corresponding to a single BS chain. The scan output of RU_i is masked via the i^{th} AND gate if the i^{th} bit of the counter equals "0". This is logically equivalent to disconnecting RU_i from the MISR.

In test mode, all chain select lines are held to a logical "1" so that the scan outputs of all RUs pass the AND gates to be compacted in the MISR. The resulting signature is only an answer to a fail/pass question. It does not identify the faulty unit(s), but instead it only indicates the presence of errors in the MUT.

For diagnostic purposes, the counter is designed to step in a specific sequence such that in any BIST session, it logically disconnects all but a single RU_i from the MISR via the controlling AND gates.

At the end of a test session, the RU_i signature is compared to its reference stored in the unit or in a storage medium within the MUT [47]. The counter contents indicate which RU is faulty so that it can be replaced or a recovery process such as retry, rollback or reconfigure can be activated in a multiprocessor system.

The hardware overhead of this diagnosable architecture includes an LFSR, the chain selection logic (counter, AND gates), a signature comparator (XOR gates), and a MISR.

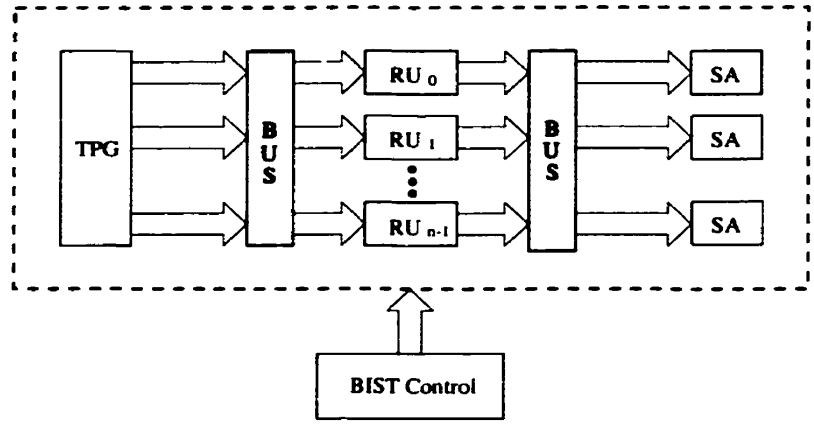


Figure 4.5: General form of a distributed BIST architecture.

This diagnostic method is time consuming because of the time it takes to shift test data in and out of the BS chains. The number of BIST sessions required to diagnose multiple faulty units is still n, even if there is a single faulty unit on the module. Clearly this diagnostic method is not very efficient when the actual number of faulty RUs is small, which is usually the case in real VLSI modules. But this along with the silicon overhead associated with a boundary scan design maybe the price to pay for simple test and diagnosis automation in systems, boards, or MCMs.

4.1.2 Distributed Diagnostic BIST Architectures

The general form of a distributed BIST architecture is shown in Figure 4.5.

In this architecture, each unit is associated with its own ORA circuitry. A single TPG or n different ones can be used. A distributed BIST provides a high degree of parallelism, since all the RUs can be tested at the same time. This leads to an increased hardware overhead, but a reduced test time, and usually more accurate diagnosis [3]. It also leads to a higher fault coverage because the TPG and ORA circuitry can be customized individually for each unit.

During testing, the BIST controller communicates with the RUs using system or test buses. It steps them through the BIST test sequence in parallel as explained next.

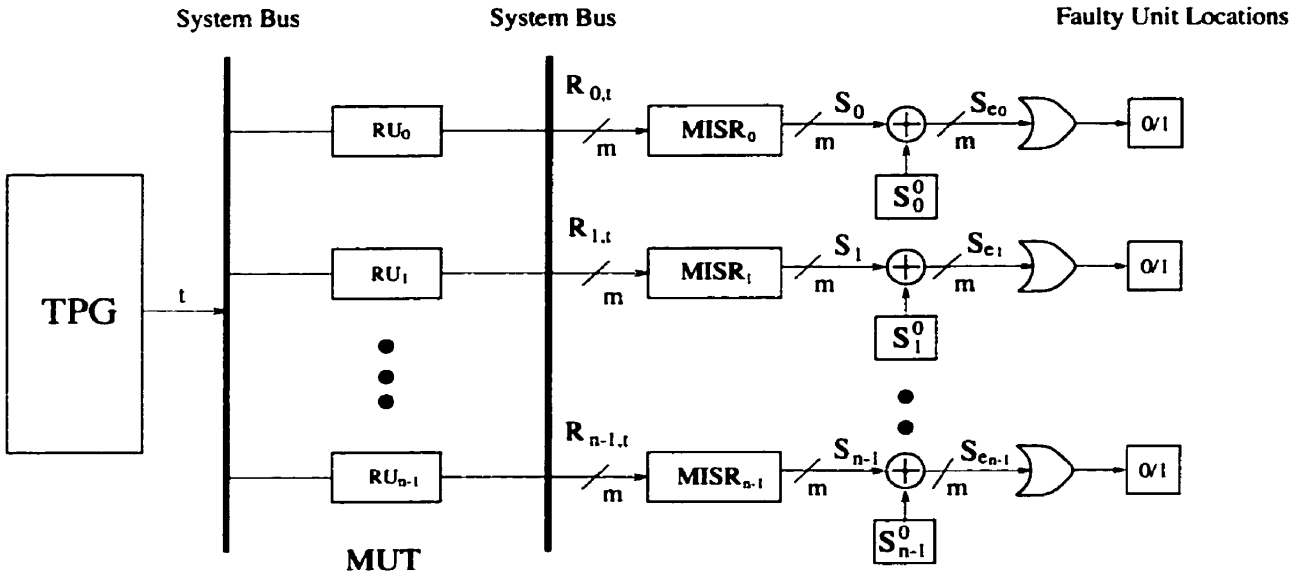


Figure 4.6: Distributed parallel diagnosis using the system bus.

Diagnosis Using the System Bus

The diagnostic architecture for the straightforward parallel approach is shown in Figure 4.6.

The diagnostic procedure is similar to the centralized serial one except that when a new test pattern is generated, it is transferred via the system bus to all of the RUs at the same time, and their test responses are simultaneously compacted in n MISRs. After compacting the responses from the last test, the error signatures $(S_{e_0}, S_{e_1}, \ldots, S_{e_{n-1}})$ are calculated in parallel. Therefore, all fail/pass bits are readily available. This allows for a quicker diagnosis, since each of these units is directly identifiable by its individual fail/pass bit, without any additional decoding.

It is clear that the distributed approach is n times faster than the centralized one, requiring only a single BIST session. But it also requires almost n times the hardware required by the serial approach. Therefore, it may be viable only for modules with a small number of units. Hence, there is a trade-off between the diagnosis time and the diagnostic hardware overhead. The higher the complexity of a unit, the lower the overhead will be.

Note that the hardware overhead can be reduced if the reference signatures are not stored. This is done in [29] for a multiprocessor system with identical units. The processors are organized logically in a tree structure. Every unit executes BIST and generates a signature. A *majority* signature is found and propagated to the root (host) processor.

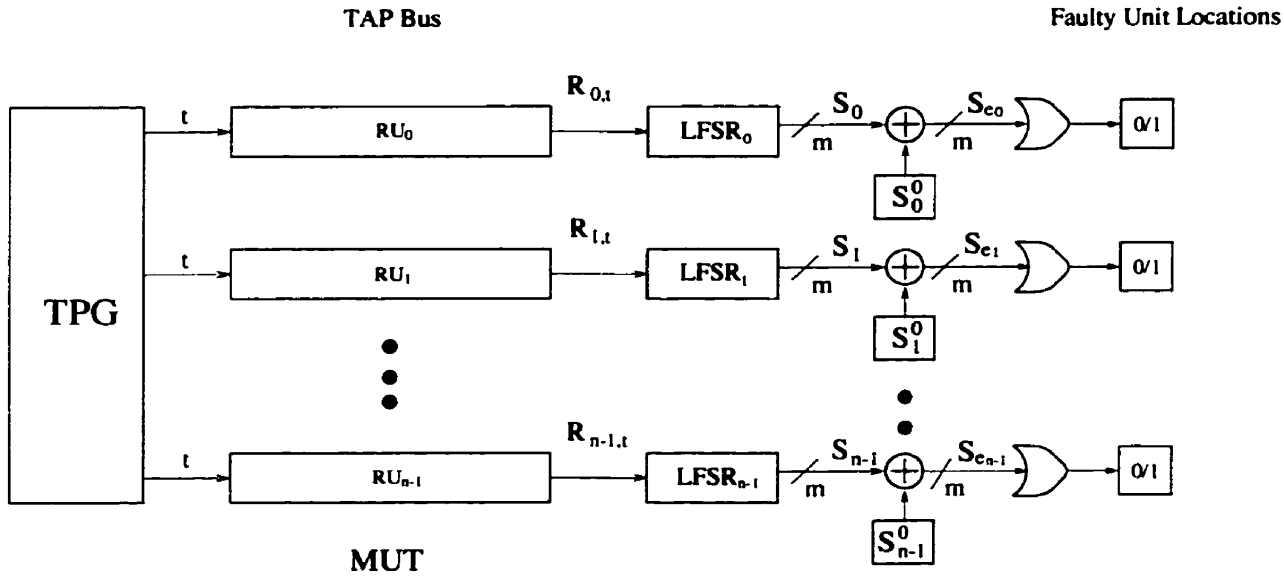


Figure 4.7: Distributed multiple BS chains architecture.

This signature is used as the *good* one, assuming that less than half of the processors are faulty. The good signature is then transferred in parallel, level by level, from the root to each unit. In parallel, each unit compares its own signature to the good one and diagnoses itself as good or faulty. Hence, the need for storing a reference signature is eliminated.

Diagnosis Using the Boundary Scan TAP Bus

A STUMPS-like module with a parallel pattern generator and a signature analyzer unique to each BS chain, provides a parallel diagnostic capability down to the RU level [132]. The multiple boundary scan architecture for distributed diagnosis is shown in Figure 4.7, where n LFSRs are used to compact the responses from each of the chains separately.

In [134], the pattern generator and signature analyzer are implemented as BILBO blocks [66] within each unit. Functional shift registers on the input side of the scan chains are configured as generating LFSRs, while those on the output side are configured as compacting LFSRs.

The internal logic is tested by running BIST for each unit in parallel. Since the RU's internal logic is isolated from the others during testing, the failing RU can be identified directly from its error signature. The test time is reduced to a single BIST session at the expense of additional hardware.

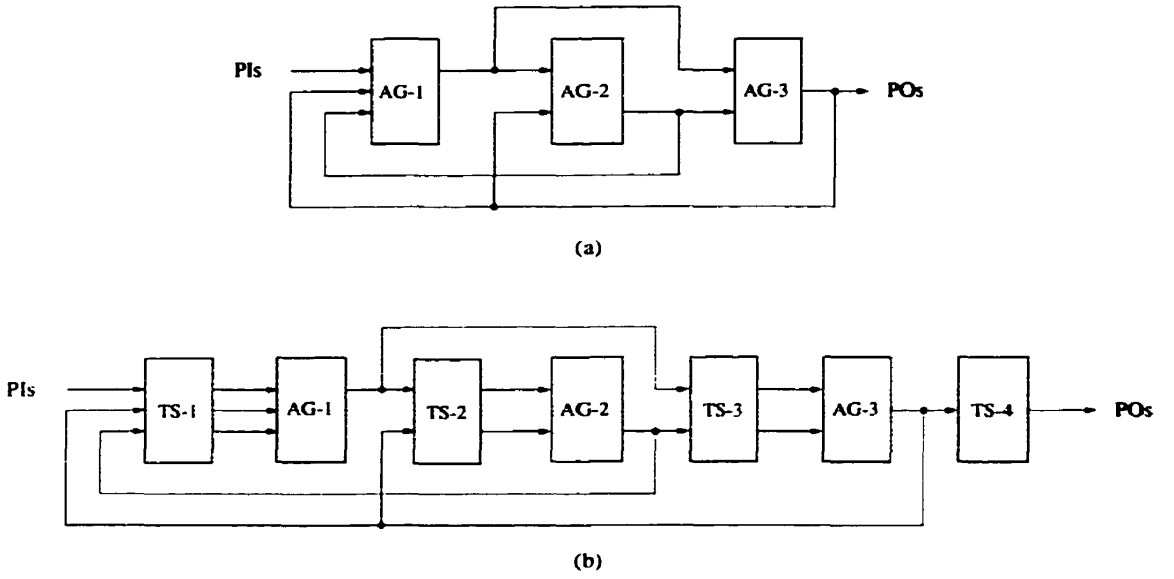


Figure 4.8: An example of test switch insertion [7].

4.1.3 Hybrid Diagnostic BIST Architectures

Fault isolation to a group of units on a module is a minimal maintenance requirement for most diagnostic applications [7]. An Ambiguity Group (AG) is defined as the smallest set of units to which a fault can be isolated [7]. In a hybrid diagnostic architecture, the RUs are partitioned into groups sharing the same BIST resources such as in the centralized architectures, with each group having its own set of BIST resources such as in the distributed architectures.

Fault isolation to an AG is done using a Test Switch (TS) [7] similar to the well-known BILBO structure. The TS generates pseudorandom test patterns for application to an AG inputs while simultaneously compacting the incoming test responses from another AG outputs. When possible, functional registers in the MUT can be configured during test to implement these switches.

After the AGs are identified, using appropriate design partitioning techniques, the test switches are added such that all inputs to an AG pass through a TS [7]. Figure 4.8(b) illustrates the addition of test switches into the module shown in Figure 4.8(a).

The AGs are tested simultaneously if possible, or sequentially using the switches. At the end of a BIST session, the compacted signatures are shifted out from the switches and checked against their references. A negative comparison isolates the faulty unit(s) to an

AG.

The hybrid diagnostic architectures with boundary scan modules follow directly from the centralized and distributed versions with multiple scan chains. The only difference is that each chain in this case corresponds to an AG. An ambiguity group is obtained as the concatenation of the individual RU boundary scan chains [118].

Having introduced the different architectures used for diagnosis, the choice is that of a designer. There is a tradeoff between the diagnosis time, with centralized diagnosis being the slowest and the least hardware consuming, and the hardware overhead, with distributed diagnosis being the fastest and the most hardware consuming architecture. A hybrid architecture is a compromise between the diagnosis time and the diagnostic hardware overhead. It reduces the diagnosis time well below what is required by a serial architecture, and at the same time it requires less overhead than a parallel architecture. However, its diagnostic resolution is poorer than either, being only to an AG.

Finally, a word about the probability of misdiagnosis. Misdiagnosis occurs when signatures in any unit or AG are aliased. As mentioned in Section 2.5.7, this probability is equal to 2^{-m} for an m-bit SAR. Assuming that the units (AGs) are independent of each other, then the probability of missing exactly i faulty units (AGs) is equal to 2^{-im}.

4.2 BIST Fault Diagnosis With Space-Time Compaction

Early research on fault detection and identification focussed mainly on utilizing error detecting-correcting codes [135, 136], drawing parallels from the well-developed theory of error control coding in Information Theory. In this approach, the MUT outputs are encoded (space compacted) into some known codes. Depending on the codes used, several faulty units or outputs of the MUT are identified.

The use of space compaction in built-in test was first proposed in [137], where parity generating trees were used as linear space compactors.

More recently, several researchers have combined space compaction with signature analysis in diagnosing the faulty outputs or units on a module. As mentioned in Section 2.6, BIST fault diagnosis of large modules with VLSI units having many outputs requires a large volume of memory for reference-signature storage. Space compaction combined with time compaction of test responses can essentially reduce the overhead required for testing

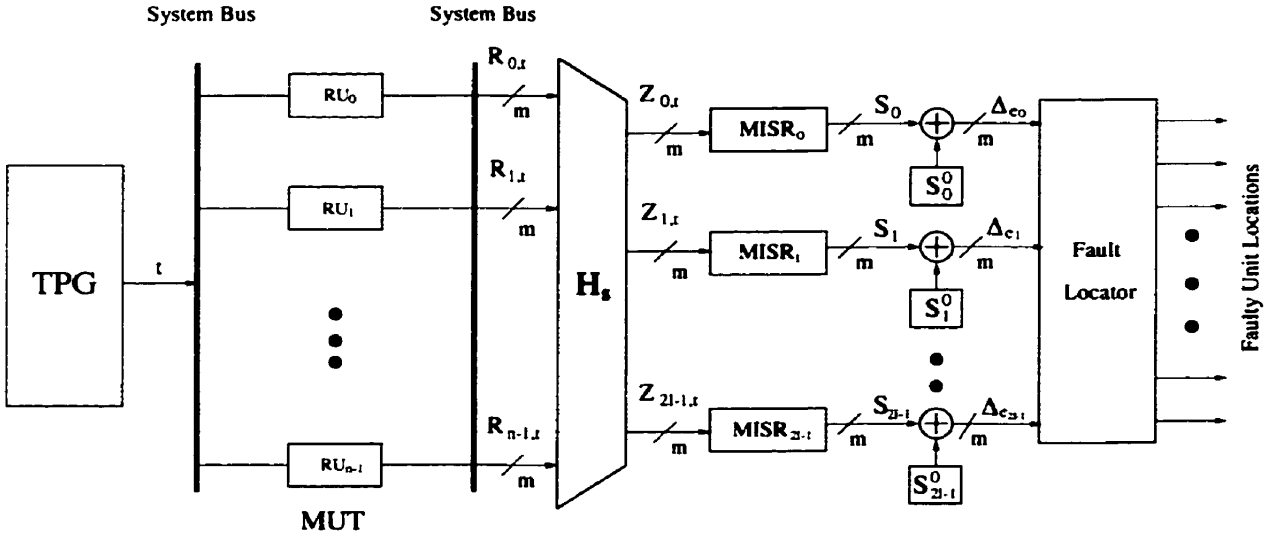


Figure 4.9: Distributed diagnosis with an STC-BIST architecture [8].

and diagnosis.

In this section, two different diagnostic architectures are presented that can be used with Space-Time Compaction (STC), outlining the benefits and downfalls of each one.

4.2.1 Distributed Space-Time BIST Architectures

The general structure of a distributed STC architecture essentially follows that of Figure 4.9 [6, 8, 34, 94, 107, 108, 138, 139].

Consider a module consisting of n units. The scheme requires all units on the MUT to be functionally interconnected through a common system bus, which is used for the transfer of test data to and from a unit. If the units are not interconnected via a common bus, an additional test bus is required for transferring the test data [6].

The proposed architecture also requires an additional diagnostic unit responsible for MUT control and the execution of the diagnostic procedure presented below. The diagnostic unit consists of four components: a pseudorandom test pattern generator implemented as an LFSR, a space-time compactor, a comparator, and a test control macro.

The width of the bus (m) is equal to the maximum number of inputs or outputs of all units on the module. It can be assumed without loosing generality that all units have the same number of outputs m. Additional zero components can be assigned to the units with

less than m outputs.

The TPG applies a total of T test patterns to all units on the MUT. The test responses can be represented by a $(n \times T)$ test response matrix $\mathbf{R}$,

$$\mathbf{R} = \begin{bmatrix} R_{0,0} & R_{0,1} & \cdots & R_{0,T-1} \\ R_{1,0} & R_{1,1} & \cdots & R_{1,T-1} \\ \vdots & \vdots & \ddots & \vdots \\ R_{n-1,0} & R_{n-1,1} & \cdots & R_{n-1,T-1} \end{bmatrix} , \tag{4.1}$$

where $R_{i,t} \in \mathrm{GF}(2^m)$, $0 \leq i < n$, and $0 \leq t < T$.

The responses of all units at time t, i.e., the t^{th} column of $\mathbf{R}$:

$$\mathbf{R_t} = \begin{bmatrix} R_{0,t} \\ R_{1,t} \\ \vdots \\ R_{n-1,t} \end{bmatrix} , \tag{4.2}$$

are sequentially entered into the space compactor via the system bus.

In error control coding, a class of codes, known as non-binary (or byte) multiple error correcting codes, is capable of locating and correcting errors in a sequence of m-bit bytes [84]. The space compactor is an application of these codes. The $(2l \times n)$ check matrix ($\mathbf{H_s}$) for a $2l$-error-detecting l-error-correcting code is given by:

$$\mathbf{H_s} = \begin{bmatrix} h_{0,0} & h_{0,1} & \cdots & h_{0,n-1} \\ h_{1,0} & h_{1,1} & \cdots & h_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{2l-1,0} & h_{2l-1,1} & \cdots & h_{2l-1,n-1} \end{bmatrix} , \tag{4.3}$$

where $h_{i,j} \in \mathrm{GF}(2^m)$, and l is the maximum number of faulty RUs on the module.

The intermediate space signatures corresponding to the t^{th} test,

$$\mathbf{Z_t} = \begin{bmatrix} Z_{0,t} \\ Z_{1,t} \\ \vdots \\ Z_{2l-1,t} \end{bmatrix} , \tag{4.4}$$

are related to $\mathbf{R_t}$ through the equation:

$$\mathbf{Z_t} = \mathbf{H_s}\mathbf{R_t} , \tag{4.5}$$

where $Z_{j,t} = \sum_{i=0}^{n-1} h_{j,i} R_{i,t} \in \mathrm{GF}(2^m)$, $0 \leq j < 2l$.

In theory, any l-byte Error Correcting (l-bEC) code can be used for space compaction. In [6, 94], l-bEC 2^m-ary Reed-Solomon (RS) codes [76, 98] were used for this purpose, with

$$\mathbf{H_s} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{2l} & \alpha^{4l} & \cdots & \alpha^{2l(n-1)} \end{bmatrix} , \tag{4.6}$$

where $n \leq 2^m - 1$, and α is a primitive in $\mathrm{GF}(2^m)$.

The generator polynomial corresponding to this matrix is equal to [84]:

$$f(x) = \prod_{i=1}^{2l} (x + \alpha^i) \ . \tag{4.7}$$

Hence, the computation of the intermediate signatures $\mathbf{Z_t}$ can be performed by $2l$ MISRs with feedback polynomials $f_i(x) = x + \alpha^i$, $1 \leq i \leq 2l$. Note that the *space* MISRs must be cleared before the individual responses of a new test are shifted in.

At this point, each of the m-bit space signatures are entered in parallel into the time compactor which consists of $2l$ identical m-bit MISRs, with feedback polynomials corresponding to α. The *time* MISRs are all initialized to zero before $\mathbf{Z_0}$ is compacted. As mentioned in Section 2.5.6, these MISRs multiply their contents by α and add their inputs.

When the last intermediate space signatures $\mathbf{Z_{T-1}}$ are compacted, the contents of the time MISRs are the space-time signatures,

$$\mathbf{S} = \begin{bmatrix} S_0 \\ S_1 \\ \vdots \\ S_{2l-1} \end{bmatrix} , \tag{4.8}$$

where,

$$S_j = \sum_{t=0}^{T-1} Z_{j,t} \alpha^{T-1-t} \tag{4.9}$$

$$= \sum_{t=0}^{T-1} \left(\sum_{i=0}^{n-1} h_{j,i} R_{i,t} \right) \alpha^{T-1-t} \ , \tag{4.10}$$

$0 \leq j < 2l$. It should be pointed out that although the above analysis uses a primitive feedback polynomial $f(x) = x + \alpha$ for the time MISRs, any polynomial that yields a satisfactory detective aliasing can be used instead.

In a matrix form, the above analysis is mathematically equivalent to computing all the intermediate space signatures first:

$$\begin{aligned} \mathbf{Z} &= \mathbf{H_s R} && (4.11)\\ &= \begin{bmatrix} \mathbf{Z_0} & \mathbf{Z_1} & \cdots & \mathbf{Z_{T-1}} \end{bmatrix} && (4.12)\\ &= \begin{bmatrix} Z_{0,0} & Z_{0,1} & \cdots & Z_{0,T-1} \\ Z_{1,0} & Z_{1,1} & \cdots & Z_{1,T-1} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{2l-1,0} & Z_{2l-1,1} & \cdots & Z_{2l-1,T-1} \end{bmatrix}, && (4.13) \end{aligned}$$

and then compacting the whole sequence in $\mathbf{Z}$ into the time MISRs, forming the final signatures $\mathbf{S}$. Thus,

$$\begin{aligned} \mathbf{S} &= \mathbf{Z H_t} && (4.14)\\ &= \mathbf{H_s R H_t}\ , && (4.15) \end{aligned}$$

where the time compaction matrix ($\mathbf{H_t}$) is the check matrix of a $[T, T-1, 2]$ RS code over GF(2^m) given by Equation 2.48. The STC of test responses can be considered as a decoding procedure for a *concatenated* code in GF(2^m) [76], where the *inner* (space compaction) code is a $[n, n-2l, 2l+1]$ RS code, and the *outer* (time compaction) code is a $[T, T-1, 2]$ RS code [6].

The final STC signatures are saved on the MUT or shifted out for off-line analysis. They are compared in parallel with their corresponding predetermined reference signatures $S_0^0, S_1^0, \ldots, S_{2l-1}^0$ to produce the $2l$, m-bit *distortion* vectors $\Delta_{e_0}, \Delta_{e_1}, \ldots, \Delta_{e_{2l-1}}$. The comparator consists of $2l \times m$ XOR gates. Thus,

$$\Delta_{\mathbf{e}} = \mathbf{S} \oplus \mathbf{S^0}\ . \qquad (4.16)$$

If $\Delta_{\mathbf{e}} = \mathbf{0}$, then the MUT is fault-free and the diagnostic procedure is complete. However, if any of the distortions $\Delta_{e_j} \neq 0$, then the presence of the faulty RU(s) is detected and the fault locator (decoder) is invoked.

Since the minimum distance of the RS code used for space compaction is $2l+1$, then up to $2l/l$ faulty units can be detected/located. Locating the faulty RUs is achieved by analyzing the distortions in the STC signatures.

Assuming that the compactors, comparators, and the fault detection circuitry are all fault free, then the error manifestations caused by faults in the RUs can be represented by an $(n \times T)$ 2^m-ary error matrix:

$$\mathbf{E} = \mathbf{R} \oplus \mathbf{R^0} \ , \tag{4.17}$$

where $\mathbf{R^0}$ is the error-free response matrix. Hence,

$$\begin{aligned} \Delta_{\mathbf{e}} &= \mathbf{H_s E H_t} && (4.18) \\ &= \mathbf{H_s S_e} \ , && (4.19) \end{aligned}$$

where

$$\mathbf{S_e} = \begin{bmatrix} S_{e_0} \\ S_{e_1} \\ \vdots \\ S_{e_{n-1}} \end{bmatrix} \tag{4.20}$$

are the time-compacted error signatures from all RUs. Neglecting the possible aliasing in the time MISRs, $S_{e_i} = 0$ if and only if RU_i does not produce any errors for all test vectors. Otherwise, $S_{e_i} \neq 0$.

Assuming that the l faulty units occur at the *unknown* locations $i_1, i_2, \ldots, i_l$, then there will be l non-zero time-compacted error signatures at these locations. To find these locations, $\mathbf{S_e}$ is considered as the error sequence of the space-compaction code which can be decoded by solving Equation 4.19. The distortion equations are thus given by:

$$\begin{aligned} \Delta_{e_0} &= h_{0,i_1} S_{e_{i_1}} \oplus h_{0,i_2} S_{e_{i_2}} \oplus \cdots \oplus h_{0,i_l} S_{e_{i_l}} && (4.21) \\ \Delta_{e_1} &= h_{1,i_1} S_{e_{i_1}} \oplus h_{1,i_2} S_{e_{i_2}} \oplus \cdots \oplus h_{1,i_l} S_{e_{i_l}} && (4.22) \\ &\vdots \\ \Delta_{e_{2l-1}} &= h_{2l-1,i_1} S_{e_{i_1}} \oplus h_{2l-1,i_2} S_{e_{i_2}} \oplus \cdots \oplus h_{2l-1,i_l} S_{e_{i_l}} \ . && (4.23) \end{aligned}$$

This system of $2l$ simultaneous linear equations over GF(2^m) can be uniquely solved for the $2l$ unknowns $S_{e_{i_1}}, S_{e_{i_2}}, \ldots, S_{e_{i_l}}$ and $i_1, i_2, \ldots, i_l$ [84, 140]. The fault locator solves these equations for the faulty RU location(s).

Note that only $2l$ distortion signatures need to be stored for any number of units on the MUT. This results in a considerable reduction in the hardware overhead (O(lm))

required for diagnostics, as compared to the straightforward time-compaction approach where separate signatures are compacted for each unit on the MUT. An additional space compaction from GF(2^m) $\rightarrow$ GF(2^b), $b < m$, will result in additional reduction of the overhead to O(lb) [6, 94].

In an operating environment, single-faulty-unit ($l = 1$) events are the most likely to occur [129, 141]. Thus, a self-diagnostic procedure capable of locating a single faulty RU will be sufficient in most cases. This can be accomplished by a very simple decoder for a single error-correcting code. In [129, 141], the linear (Hamming) code is used for this purpose. The check matrix for this code is given by:

$$H_s = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \end{bmatrix} . \tag{4.24}$$

The application of single error-correcting RS codes for ROM testing was considered in [142]. In these cases, it is clear that STC has a great advantage over the straightforward time-compaction methods in reducing the number of reference signatures and comparisons from n to just two.

In addition, for the case of double faulty units ($l = 2$), efficient implementations are possible due to the simple analytical solutions of Equation 4.19 [6].

However, it is clear that as l increases above two, the analytical solutions become very cumbersome, requiring a complex decoding procedure that is difficult to implement in hardware. Consequently, a diagnostic algorithm is presented in [6] that utilizes a modified version of the Euclidean algorithm [143] as a recursive procedure for calculating the faulty RU location(s).

In a multiprocessor environment [8, 138, 139], another alternative to implementing the decoding procedure is an Assembly language program that utilizes the internal microprocessor architecture as efficiently as possible [144, 145]. The execution of such a program on an m-bit microprocessor assuming a CPI (Clocks Per Instruction) of one, will require an additional O(lT) time overhead as compared to a VLSI hardware implementation. Since the number of pseudorandom tests can be quite large, this alternative may not be a viable option for large multiprocessor systems.

Finally, it should be mentioned that the diagnosis theory presented in this section can be extended directly to MUTs utilizing cellular automata for space-time compaction. In [34], the output responses of the MUT are encoded by applying the encoding strategy of a CA-based bEC code. The encoded symbols for different test patterns are compacted into STC

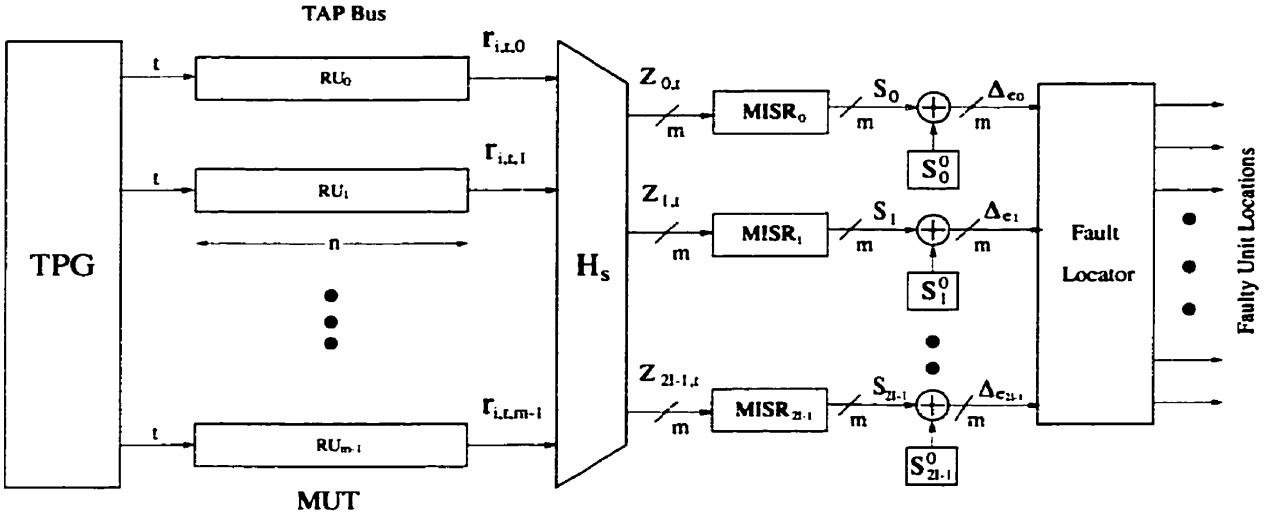


Figure 4.10: Distributed diagnosis with an STC-STUMPS architecture.

signatures. Employing the same decoding structure of the bEC code, the faulty units can be detected and located [34]. The TPG can also be implemented using a CA structure. The use of CA-based STC structures exploit the inherent advantages of regularity, modularity, and cascadability of CA, which is well suited for VLSI implementations.

Diagnosis Using the Boundary Scan TAP Bus

The distributed STC architecture can be extended directly to modules employing the boundary scan TAP bus. Assuming a STUMPS-like multiple scan structure, the STC-BIST architecture for distributed diagnosis is shown in Figure 4.10.

As before, the proposed architecture requires an additional diagnostic unit responsible for MUT control and the execution of the diagnostic procedure. The necessary control signals for the TAPs are also provided by the on-module diagnostic unit.

There are m scan chains each corresponding to a single RU. Without loss of generality, it is assumed that all chains are of equal length (n).

Define a *scan cell frame* (F_i, $0 \leq i < n$) to be the set of m scan flops containing the i^{th} cells from all scan chains [38]. A formal definition is given in Chapter 5.

The t^{th} test response captured in all frames, $F_{i,t} \in \text{GF}(2^m)$, are shifted sequentially into the space-time compactor via the TAP bus while a new test is shifted-in from the TPG.

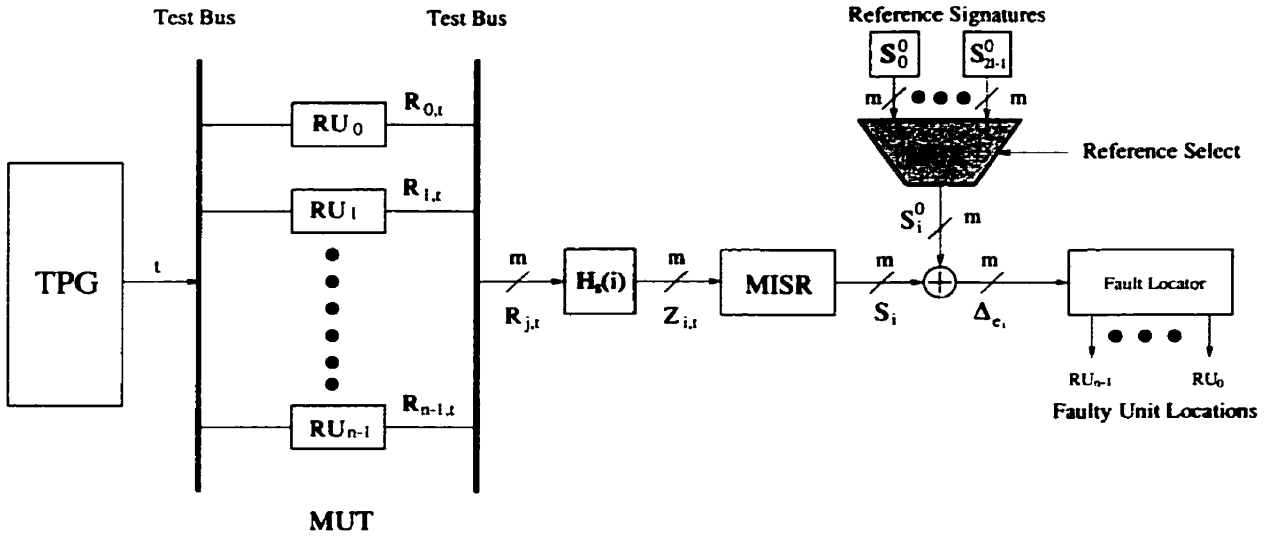


Figure 4.11: Centralized diagnosis with an STC-BIST architecture.

It should be clear at this point that diagnosing a STUMPS module is identical to the one employing the system bus, with the scan frames $F_{i,t}$ corresponding to the RU outputs $R_{i,t}$. A faulty scan frame is one which captures errors during test. Hence, the STC diagnostic procedure in a STUMPS module can locate up to l faulty frames, without knowing exactly which scan cells fail.

To improve the diagnostic resolution, a simple approach is to diagnose one chain at a time. This can be accomplished by adding a chain selector such as the one shown in Figure 4.4 to selectively gate the scan-out data. This scheme requires repeating the BIST session m times in order to diagnose all scan chains.

In each test session, the complete test set is applied to all chains, but only the test responses from scan chain j $(0 \leq j < m)$ are fed into the diagnostic unit. Clearly, the correct identification of the failing scan cell(s) is guaranteed since each frame in this case corresponds to a single cell.

4.2.2 Centralized Space-Time BIST Architectures

The general structure of a centralized STC architecture essentially follows that of Figure 4.11.

Consider again a MUT consisting of n units interconnected through a common m-bit

system bus, or a STUMPS-like MUT with m BS chains each having n scan cells. As before, an additional diagnostic unit is responsible for MUT control and the execution of the diagnostic procedure. The diagnostic unit consists of a parallel pseudorandom test pattern generator, a *sequential* space compactor, a *single* time compactor, a comparator, and a test control macro.

The centralized scheme is *theoretically* identical to the distributed one, except that each of the $2l$ STC signatures are computed serially one after the other. For each signature, the space compactor is set to implement the i^{th} row of $\mathbf{H_s}$:

$$\mathbf{H_s}(i) = \begin{bmatrix} h_{i,0} & h_{i,1} & \cdots & h_{i,n-1} \end{bmatrix} , \tag{4.25}$$

where $h_{i,j} \in \text{GF}(2^m)$, and $0 \leq i < 2l$.

Assuming that the test set consists of T test vectors, then the intermediate space signatures corresponding to $\mathbf{H_s}(i)$ are given by

$$\mathbf{Z}(i) = \mathbf{H_s}(i)\mathbf{R} . \tag{4.26}$$

In reality, each intermediate space signature is time-compacted into an m-bit MISR as soon as it is formed in the space compactor. However, it is mathematically equivalent to computing all the intermediate signatures first, and then compacting the whole sequence in $\mathbf{Z}(i)$ into the MISR, forming the final STC signature

$$S_i = \mathbf{H_s}(i)\mathbf{R}\mathbf{H_t} . \tag{4.27}$$

The signature S_i is compared to its reference S_i^0, thus forming the distortion vector Δ_{e_i}, which is saved on the MUT or shifted out for off-line analysis. Subsequently, the space compactor is set to implement another row of $\mathbf{H_s}$, and the whole process is repeated until all of the $2l$ STC signatures are collected. At that point, the diagnostic procedure used with the distributed architectures can be applied directly to locate the faulty units or faulty scan frames.

The use of a single sequential space compactor and a single MISR leads to a reduced hardware overhead, when compared to a distributed architecture, at the expense of increased diagnosis time for repeating the same BIST session $2l$ times. Compared to the hardware overhead, which imposes recurring silicon cost for every MUT, the test time expenditure is only a single time cost for a few faulty MUTs that require diagnosis [38].

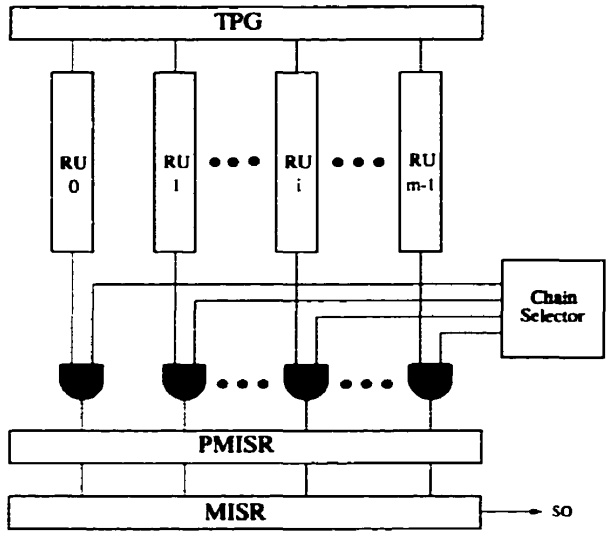


Figure 4.12: A high resolution centralized STC-STUMPS architecture.

The use of a centralized STC-STUMPS architecture is presented in [38]. In that work, the sequential space compactor is implemented as the check matrix of the aforementioned l-bEC RS code with generator polynomial $f(x) = (x + \alpha)(x + \alpha^2) \cdots (x + \alpha^{2l})$. The space compactor can thus be implemented with a Programmable MISR (PMISR), having a programmable feedback polynomial $f_i(x) = x + \alpha^i$, $1 \leq i \leq 2l$, corresponding to each row in $\mathbf{H}_s$.

Up to l faulty scan frames can be located with this method. As mentioned in Section 4.2.1, a chain selector can be used to improve the diagnostic resolution as shown in Figure 4.12 [38]. However, this requires repeating the BIST session $2ml$ times in order to diagnose all scan cells.

This excessive test time overhead can be reduced at the expense of additional hardware, by using multiple sets of the diagnostic hardware [38]. This results in a hybrid architecture similar to the one used with time compaction. Multiple copies of the diagnostic hardware can be used as long as the hardware overhead is acceptable. When $2l$ copies are used, the method in [38] reduces to the one described in Section 4.2.1.

Although the method can theoretically locate up to l faulty scan frames or l faulty cells in each chain, in practice, it is usually limited to only ($l = 1$). This is because a generic equation solver for Equation 4.19 is hard to develop.

Finally, it should be mentioned that another version of an STC-STUMPS architecture

is presented in [37] for MCMs. In that work, it is assumed that faults can occur in only a single chip. Thus, there is no need for a scan chain selector. The space compactor is implemented as the check matrix of a single error detecting and correcting Hamming code [76]. The time compactor on the other hand is implemented as a MINFSR. The diagnostic unit is implemented as a Field Programmable Gate Array (FPGA), which can also be incorporated on the MCM [37].

Chapter 5

The GSTUMPS Architecture

The adopted BIST architecture to be used with the proposed diagnostic algorithms in the thesis is a generalized version of the STUMPS architecture [146]. The Generalized STUMPS (GSTUMPS) architecture is a generic BIST structure that can be used at the chip, MCM, board, or system level. An illustration of the global structure of GSTUMPS is shown in Figure 5.1.

The MUT latches or flip-flops, are designed in conjunction with the LSSD or scan-path design methodologies. Boundary scan is also applied, so that all but the embedded RAMs can be tested as combinational circuits. Henceforth, each scan state variable will be referred to as a shift register stage or a scan cell.

The scan chains that connect between the SRSG and MISR are known as channels. If the MUT is a system, then each channel represents a BS chain of either a board, a processor or an MCM. If the MUT is a board, then each channel represents a BS chain of the chips mounted on it. If the MUT is an MCM, then the channels represent the BS chains of the chips mounted on the common substrate, and in addition, it might contain the internal scan chains of some or all of the chips. Finally, if the MUT is a chip, then one of the channels represent the BS chain, and the rest are the chip's internal scan chains.

Assume that the MUT has m channels. Although the number of scan cells in each channel may vary, there is a test application time advantage in structuring the channels so that they each contain roughly the same number of scan cells [1].

The Parallel Shift-Register Sequence Generator (SRSG) is a combination of an LFSR and a phase-shifting XOR network used to generate the pseudorandom test patterns. The

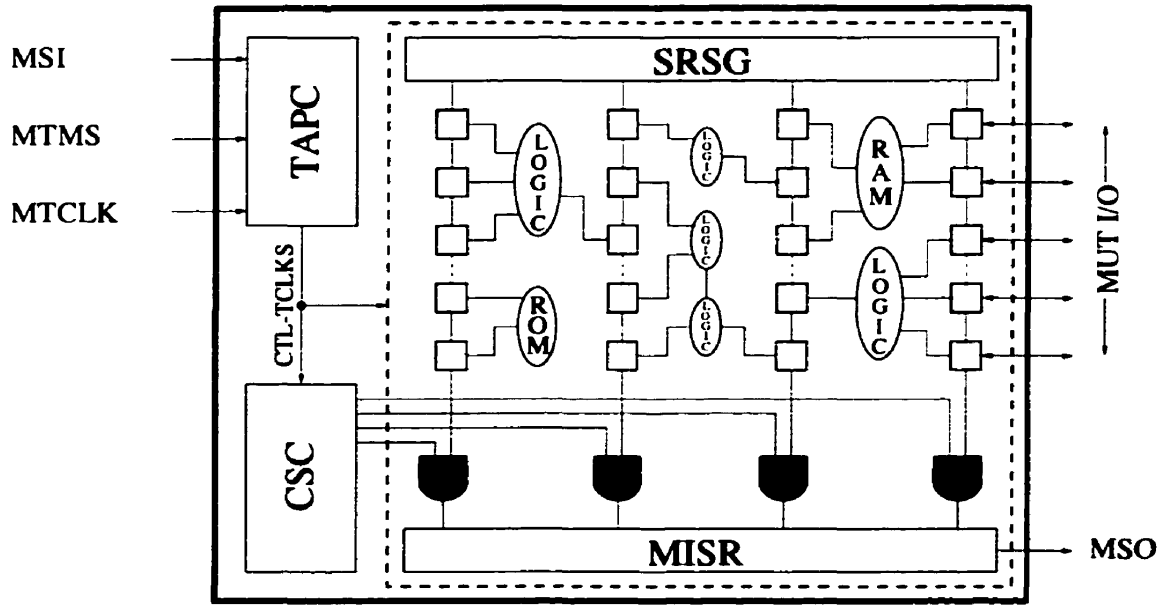


Figure 5.1: GSTUMPS: A global BIST structure.

SRSG contains a SRS for each channel on the MUT. The XOR network provides shifted versions of the patterns generated by the LFSR. The use of this network avoids the possible test pattern degradation that occurs when the outputs of the SRSG stages feed the channels directly [74].

The BIST compactor is a conventional m-bit MISR. The scan cells shifting clocks and the shift clocks of both the SRSG and the MISR are tied together.

The Test Access Port Controller (TAPC) supplies the signals for controlling the MUT during test. It includes the the boundary scan IRs and is also implemented with SRSs.

The GSTUMPS architecture is specially tailored for diagnosis. There are different diagnostic demands placed on the MUT at different packaging levels. The first step in diagnosing the MUT is to locate the Faulty Channel(s) (FC(s)). If the MUT is a system, then identifying the FC(s) will in turn identify the faulty board(s), MCM(s) or processor(s) and the diagnosis is complete. If MUT is a board or an MCM, then again the FC(s) will identify the faulty chip(s) and no further diagnosis is required. Finally at the chip level, identifying the faulty channels is the first step in diagnosing it.

To meet these objectives, a Channel Select Controller (CSC) is used to gate the channels outputs such that a single/multiple scan channel(s) is/are made observable to the MISR during diagnostic testing. The CSC, discussed in Chapters 6 and 7, is implemented with

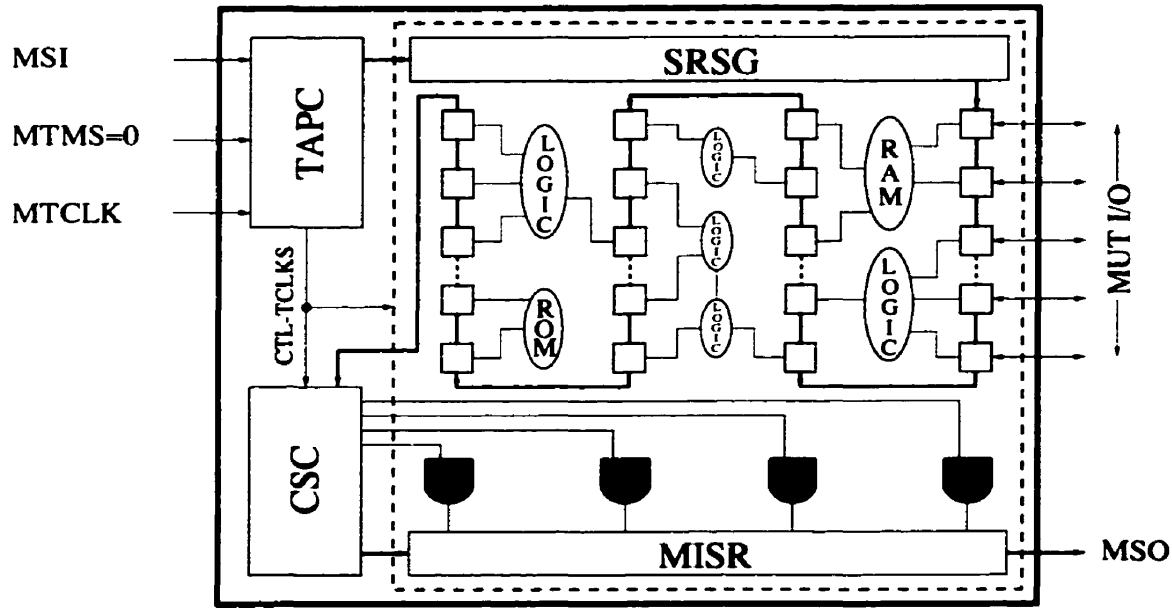


Figure 5.2: GSTUMPS configuration during normal MUT operation.

SRSs.

Neither the SRSG nor the MISR is a part of the MUT logic. Separate test macros contain the SRSG, MISR, TAPC, CSC, and their switching and control circuitry. The test macros are wired only into the scan paths of the MUT, and thus they have a negligible performance effect on the critical paths of the MUT. These macros can be overlayed on an already completed design.

The GSTUMPS configuration allows for BIST testing and diagnosis with a minimal control and clocking from an external test system. It requires at least four additional test pins: two Master Scan I/O pins (MSI, MSO), a Master Test Clock pin (MTCLK), and a Master Test Mode Select pin (MTMS). Additional pins maybe used to reduce the test and diagnosis time by loading/unloading data from/to the tester in parallel. As will be explained in Chapter 8, the digital tester can be very simple.

During the normal *scan-mode* (MTMS = 0), test circuitry does not affect the MUT's operation. All scan cells, including the ones in the SRSG, MISR, CSC and TAPC, are configured as a single shift-register chain connected between the MSI and MSO pins. This is shown in Figure 5.2.

In BIST-mode (MTMS = 1), the scan-in and scan-out ports of each of the m channels are connected to an individual SRSG and MISR stage respectively. The basic structure of

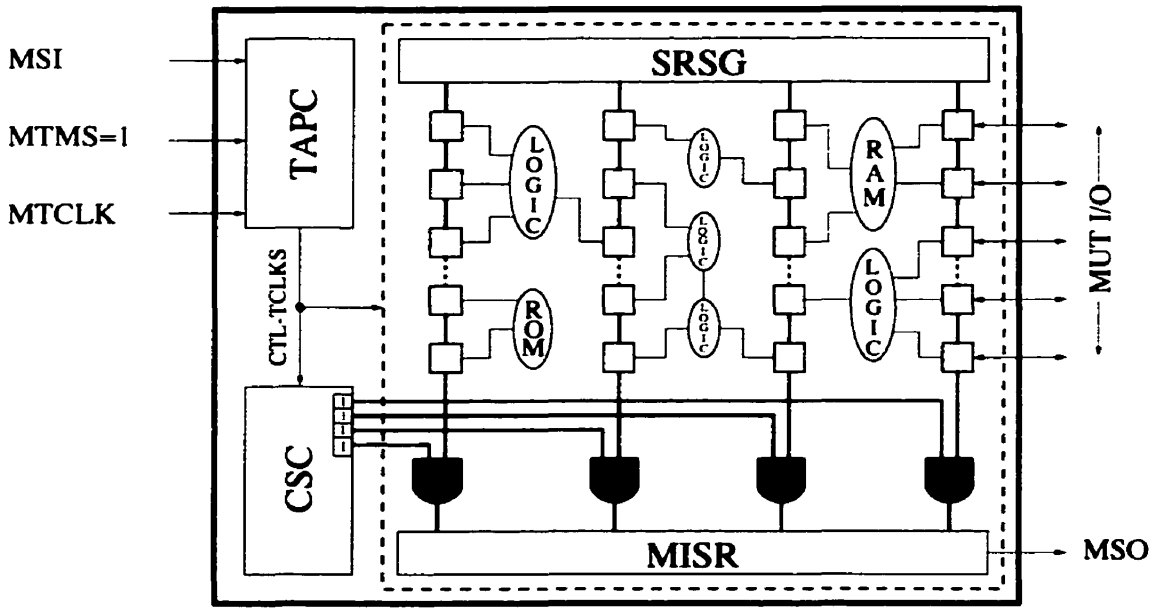


Figure 5.3: GSTUMPS configuration during BIST.

the MUT acting in the self-test mode is shown in Figure 5.3.

Using a shift clock, the channels are loaded with pseudorandom test patterns from the SRSG. The number of scan clock cycles required to fill the channels is equal to the number of scan cells in the longest channel. The data from shorter channels merely overflows into the MISR where it becomes a part of the signature.

Once the stimuli from the SRSG is shifted into all SRSs of all channels, the MUT's system clock is applied, thus capturing the test results back into the SRSs. These are then scanned out into the MISR, while simultaneously loading the next test pattern set from the SRSG. After compacting the responses from the last test, the accumulated signature in the MISR is unloaded into the tester for off-line analysis. Whether the MUT is faulty or not is determined by comparing the compacted signature with the expected fault-free one.

Note that the MUT must be initialized to a repeatable (not necessary known) state before starting a BIST session. If the channels are not initialized, the non-repeatable state of the SRSs at power-on will be scanned into the MISR on the first test, thus corrupting the final signature.

The key to initialization is to be able to repeat the identical initialization patterns each time BIST is invoked. An acceptable initialization procedure is to apply (in scan-mode)

predefined logic values on the MSI and run enough scan clock cycles to load the channels with these values.

The MISR may be initialized with any pattern (usually all zeros), but the SRSG must be loaded with a nonzero seed as explained before in Section 2.3.2.

Embedded RAMs are initialized by applying several SRSG pseudorandom patterns while the MUT is acting in BIST-mode. Once, the initialization of all memory cells is completed, the MISR must be reinitialized because of the unpredictable values which have been shifted in during the initialization phase. This re-initialization can be avoided by turning-off all of the CSC's channel select lines during the initialization process.

5.1 Validating the GSTUMPS Test Circuitry

An important issue in BIST utilization for fault detection and diagnosis is the testing of the BIST circuitry itself. A common assumption is that the BIST circuitry has a smaller failure rate compared to the MUT [7]. Although it maybe a true assumption, it is not sufficient to guarantee a high quality test or diagnosis. The design of GSTUMPS addresses this issue.

Several tests are used to validate the test circuitry in GSTUMPS. They can be grouped under scan-cell testing, CSC testing, and TAPC-SRSG-MISR testing. The BIST facilities as well as clocks and other stimuli from an inexpensive external test system are used during these tests.

5.1.1 Validating the Scan Chains

Several scan-based test schemes are used in the industry. The majority of these schemes assume that the scan chains are fault-free in order to test or diagnose faults in the remaining MUT logic. However, this is often not the case [54]. Therefore, such testing schemes fail in the presence of faults in the scan chains.

The circuitry associated with the scan chains may occupy a significant area, as much as 30% of a module's area [54, 147], and should not be neglected during testing. Thus, it is necessary to device mechanisms to test and diagnose faults in the scan chains themselves [54, 118, 147, 148, 149]. The scan chains are tested first to make sure that test patterns and test responses are correctly transmitted and received from each channel.

During a scan-cell test, the MUT is configured in scan-mode resulting in the single master chain configuration shown in Figure 5.2. The purpose of this test is to verify the complete connection from the MSI pin to the MSO pin, the switching capability of the scan-data portion of each scan cell, and the proper operation of the clock input(s) to the SRSs. In addition, the ability to switch between scan and BIST modes is checked.

A fault of a stuck-at nature can be easily detected by applying a sequence of 0s and 1s to the master scan chain. The test set consists of two test protocols: FLUSH and SCAN [118].

The FLUSH test places a 1 on the MSI pin and then turns-on and holds the scan shift clock(s), causing all the scan cells to flush the 1 through the master chain. The outputs at the MSO pin are verified, the MSI is switched to 0, and the test is repeated. Finally, the MSI input is switched back to 1 and the MSO output is verified once more.

This sequence of tests demonstrates the SRSs ability to flush both 1s and 0s through the scan channels, and that all the channel-to-channel connections exist [118]. For FLUSH test failures, the MUT typically has an open in the master chain due to a clock control, scan cell, or a channel-to-channel connection problem.

The SCAN test consists of loading the test pattern "$t = \ldots 00110 \ldots$" at the MSI input and scanning it through the master chain via successive scan clocks. The test data is constructed such that all possible transitions (00, 01, 11 and 10) occur in each scan cell. The MSO output is checked to ensure that the correct pattern is scanned all the way through the chain [118]. Since the total number of scan cells is known, the single or multiple faulty scan cells can be detected by inspecting the scan-out sequence.

For SCAN test failures which pass the FLUSH test, the problem is typically a stuck-on clock [118]. This allows the test stimuli to be flushed through the chain, but the SCAN test fails due to the test bits being flushed through multiple scan cells when they are supposed to be shifting one cell at a time. This causes the master chain to appear shorter than it would be with a *good* scan clock.

Scan chain defects are diagnosed using specialized diagnostic tests [54, 118, 147, 148, 149]. The diagnostic data, collected while the GSTUMPS module is in the test socket, is shifted into a diagnostic module on the external test system where it is stored for off-line analysis. The data consists of one bit per scan cell per test pattern. The analysis information determines the specific point(s) in the channel(s) where the diagnostic data changes to a solid stuck-at value.

5.1.2 Validating the CSC

Next, the CSC is validated. This is also done while the module is in scan-mode. A channel select combination is shifted into the CSC scan register through the master scan chain. The channels are initialized with alternating 0s and 1s such that alternate patterns of 0s and 1s are present at the AND gate inputs driving the MISR. For each channel select combination, the values driven into the MISR are scanned out and checked for correctness. If a failure is observed, then the problem is typically faulty AND gates, or/and faulty scan cells in the CSC.

5.1.3 Validating the TAPC, SRSG and MISR

Finally, the module is configured in BIST-mode. The SRSG and MISR are initialized as described previously with the MISR inputs being gated-off. The TAPC is put in a Validate-BIST mode through the MTMS and MTCLK pins. Once the initialization is done, the MISR inputs are gated-on and the test patterns from SRSG are shifted into the MISR using successive scan clocks. A correct signature ensures that the BIST configuration is present and is functioning properly.

5.2 Test System Requirements for GSTUMPS

In the following, the general characteristics of the external test system supporting the testing and diagnosis of failures in GSTUMPS modules are summarized.

The test system for a GSTUMPS package must be able to power the package, initialize it, select the test mode, select the test patterns (either from ATPG tools or the SRSG on the module), and *optionally* supply the channel select values during test or diagnosis.

It must also count tests (go to test t, backtrack, or stop at test t), clock the package (system and scan clocks) to complete the test, unload MISR signatures, and make pass/fail decisions.

A key feature of any test system is the software supporting it. Clearly there is a software system that controls the tester's internal execution [1]. The software for test system support manages the flow of data between the different sources. This includes the package data to

the tester, the results data from the MUT after a test is completed, the status data about the package and/or the tester, and the failure analysis data to the MUT-history database.

The software system also contains various test-generation, diagnosis, and failure analysis modules (programs). These will be revisited later in Chapter 8.

5.3 Theory and Operation of GSTUMPS

In this section, the empirical relations between the different signatures in a GSTUMPS module are developed. These relations are critical in the development of the diagnostic algorithms proposed in Chapters 6 and 7. They minimize the number of signatures that must be collected during diagnosis, thus making these algorithms practically feasible. In the following, two modes of compaction are considered for the MISR in the GSTUMPS module: the Multiple Scan Channel (MSC) mode and the Single Scan Channel (SSC) mode.

5.3.1 Signature Relations in MSC Compaction Mode

Consider first the normal parallel compaction from all channels in a GSTUMPS module. In this mode, the CSC enables/disables the inputs to the MISR, by ANDing the channel outputs with one/zero. The matrix formulation developed in Section 2.5.6 will be followed throughout this section.

Without loss of generality, assume that all m scan channels are of equal length (n).

Definition 5.3.1 *A Scan Cell Frame is the set of m scan cells containing the i^{th} cell from all scan channels [38].*

Assuming that the test set consists of T test patterns, then the captured test responses can be represented by an $(n \times T)$ test response matrix $\mathbf{R}$,

$$\mathbf{R} = \begin{bmatrix} R_{0,0} & R_{0,1} & \cdots & R_{0,T-1} \\ R_{1,0} & R_{1,1} & \cdots & R_{1,T-1} \\ \vdots & \vdots & \ddots & \vdots \\ R_{n-1,0} & R_{n-1,1} & \cdots & R_{n-1,T-1} \end{bmatrix}, \tag{5.1}$$

where $R_{i,t} \in \mathrm{GF}(2^m)$ represents the t^{th} test response captured in scan frame i.

Using Equation 2.32,

$$R_{i,t} = \left[\begin{array}{cccc} r_{i,t,0} & r_{i,t,1} & \cdots & r_{i,t,m-2} \quad r_{i,t,m-1} \end{array} \right] , \tag{5.2}$$

where $r_{i,t,j}$ is the output response corresponding to test t, captured in the i^{th} cell of scan channel j. The MUT responses from all frames at test t, i.e., the t^{th} column of $\mathbf{R}$, are shifted sequentially into the MISR, starting with scan frame 0, while a new test is shifted in from the SRSG.

Defining the test response captured in the j^{th} scan cell within a frame as:

$$R_{i,t,j} = \left[\begin{array}{ccccc} 0 & \cdots & r_{i,t,j} & \cdots & 0 \end{array} \right] , \tag{5.3}$$

then,

$$R_{i,t} = \sum_{j=0}^{m-1} R_{i,t,j} \ . \tag{5.4}$$

Theorem 5.3.1 *The final signature S_j compacted by an m-bit MISR for any GSTUMPS channel j while shifting zeros in the remaining ones is equal to:*

$$S_j = S_{j,-1} C^{nT} + \sum_{t=0}^{T-1} \sum_{i=0}^{n-1} R_{i,t,j} C^{(T-t)n-1-i} , \tag{5.5}$$

where T is the number of applied test patterns, n is the length of the scan channel, and C is the state transition matrix of the MISR with initial state $S_{j,-1}$.

Proof:

The value of the signature $S_{j,l}$ after compacting the l^{th} test pattern response can be

obtained recursively as follows:

$$S_{j,0} \overset{(a)}{=} S_{j,-1}C^n + \sum_{i=0}^{n-1} R_{i,0,j}C^{n-1-i} \tag{5.6}$$

$$S_{j,1} \overset{(b)}{=} S_{j,0}C^n + \sum_{i=0}^{n-1} R_{i,1,j}C^{n-1-i} \tag{5.7}$$

$$= S_{j,-1}C^{2n} + \sum_{i=0}^{n-1} \left[R_{i,0,j}C^{2n-1-i} + R_{i,1,j}C^{n-1-i}\right] \tag{5.8}$$

$$S_{j,2} \overset{(c)}{=} S_{j,1}C^n + \sum_{i=0}^{n-1} R_{i,2,j}C^{n-1-i} \tag{5.9}$$

$$= S_{j,-1}C^{3n} + \sum_{i=0}^{n-1} \left[R_{i,0,j}C^{3n-1-i} + R_{i,1,j}C^{2n-1-i} + R_{i,2,j}C^{n-1-i}\right] \tag{5.10}$$

$$\vdots$$

$$S_{j,l} \overset{(d)}{=} S_{j,-1}C^{(l+1)n} + \sum_{i=0}^{n-1} \left[\sum_{t=0}^{l} R_{i,t,j}C^{(l+1-t)n-1-i}\right] , \tag{5.11}$$

where,

(a) follows directly from Equation 2.40,

(b) follows again from Equation 2.40 with $S_{j,0}$ being the initial state of the MISR,

(c) follows similarly to (b), and finally,

(d) follows from the recursive relations in (a), (b), and (c).

Hence, the final signature after compacting the responses from the last test ($l = T - 1$) is equal to:

$$S_j = S_{j,-1}C^{nT} + \sum_{i=0}^{n-1} \left[\sum_{t=0}^{T-1} R_{i,t,j}C^{(T-t)n-1-i}\right] \tag{5.12}$$

$$= S_{j,-1}C^{nT} + \sum_{t=0}^{T-1}\sum_{i=0}^{n-1} R_{i,t,j}C^{(T-t)n-1-i} , \tag{5.13}$$

and the theorem is proved □.

Theorem 5.3.2 *The final signature S compacted by an m-bit MISR in a GSTUMPS module is equal to the module-2 summation of all scan channel signatures S_j:*

$$S = \sum_{j=0}^{m-1} S_j , \tag{5.14}$$

where $S_{-1} = \sum_{j=0}^{m-1} S_{j,-1}$ is the initial state of the MISR.

Proof:

The value of the signature S after compacting the l^{th} test pattern response can be obtained recursively as follows:

$$S_0 \overset{(a)}{=} S_{-1}C^n + \sum_{i=0}^{n-1} R_{i,0}C^{n-1-i} \tag{5.15}$$

$$S_1 \overset{(b)}{=} S_0C^n + \sum_{i=0}^{n-1} R_{i,1}C^{n-1-i} \tag{5.16}$$

$$= S_{-1}C^{2n} + \sum_{i=0}^{n-1} \left[R_{i,0}C^{2n-1-i} + R_{i,1}C^{n-1-i}\right] \tag{5.17}$$

$$S_2 \overset{(c)}{=} S_1C^n + \sum_{i=0}^{n-1} R_{i,2}C^{n-1-i} \tag{5.18}$$

$$= S_{-1}C^{3n} + \sum_{i=0}^{n-1} \left[R_{i,0}C^{3n-1-i} + R_{i,1}C^{2n-1-i} + R_{i,2}C^{n-1-i}\right] \tag{5.19}$$

$$\vdots$$

$$S_l \overset{(d)}{=} S_{-1}C^{(l+1)n} + \sum_{i=0}^{n-1} \left[\sum_{t=0}^{l} R_{i,t}C^{(l+1-t)n-1-i}\right] \ , \tag{5.20}$$

where,

(a) follows directly from Equation 2.40,

(b) follows again from Equation 2.40 with S_0 being the initial state of the MISR,

(c) follows similarly to (b), and finally,

(d) follows from the recursive relations in (a), (b), and (c).

Hence, the final signature after compacting the responses from the last test ($l = T-1$) is equal to:

$$S = S_{-1}C^{nT} + \sum_{i=0}^{n-1} \left[\sum_{t=0}^{T-1} R_{i,t}C^{(T-t)n-1-i}\right] \tag{5.21}$$

$$\overset{(a)}{=} \left[\sum_{j=0}^{m-1} S_{j,-1}\right] C^{nT} + \sum_{t=0}^{T-1}\sum_{i=0}^{n-1} \left[\sum_{j=0}^{m-1} R_{i,t,j}\right] C^{(T-t)n-1-i} \tag{5.22}$$

$$= \sum_{j=0}^{m-1} \left[S_{j,-1}C^{nT} + \sum_{t=0}^{T-1}\sum_{i=0}^{n-1} R_{i,t,j}C^{(T-t)n-1-i}\right] \tag{5.23}$$

$$\overset{(b)}{=} \sum_{j=0}^{m-1} S_j \ , \tag{5.24}$$

where,

(a) follows from the theorem's assumption for S_{-1} and Equation 5.4, and

(b) follows from Theorem 5.3.1. Hence, the theorem is proved □.

In principle, initializing the MISR to different states when compacting the channel signatures, affects only the total signature mathematics by introducing the additional bitwise-XOR operations required to initialize the MISR to the state $S_{-1} = \sum_{j=0}^{m-1} S_{j,-1}$.

However, initializing the MISR to the all zeros state before compacting any signature, simplifies the analysis and saves the extra XOR operations at the expense of only *resetting* the MISR before compaction. The resetting function is readily available in the GSTUMPS environment. Henceforth, it will be assumed that the MISR is always reseted before the start of any compaction process.

Corollary 5.3.1 *The total signature for any subset* (Ω) *of channels is equal to:*

$$S_\Omega = \sum_{j \in \Omega} S_j \ , \tag{5.25}$$

where the MISR is reseted before compacting the channel signatures.

The corollary follows directly from Theorem 5.3.2 with the extra condition on the initial states of the MISR. This corollary is instrumental in formulating the diagnostic algorithms presented in this thesis, as will be shown later in Chapters 6 and 7.

5.3.2 Signature Relations in SSC Compaction Mode

Consider next the SSC compaction mode from a single channel in the GSTUMPS module. In this mode, all inputs to the MISR are forced to zero by the CSC, except for the input from scan channel j. This input is also selectively enabled/disabled by the CSC during each scan shift operation.

Thus, in this mode, the MISR degenerates to an equivalent LFSR with divisor polynomial $G_j(x)$. Clearly, $G_j(x)$ is a function of both $G(x)$ and the exact position of the input channel j. For example, $G_0(x) = G(x)$. The polynomial formulation developed in Section 2.5.5 will be followed throughout this section.

Definition 5.3.2 *A Scan Channel Frame is the set of n scan cells contained in a GSTUMPS channel.*

Assuming again that the test set consists of T test patterns, then the test responses captured at the output of channel j can be represented by the test response sequence R_j,

$$R_j = \{R_{0,j}, R_{1,j}, R_{2,j}, \ldots, R_{T-2,j}, R_{T-1,j}\} \ , \tag{5.26}$$

where $R_{t,j} \in \mathrm{GF}(2^n)$ represents the t^{th} test response sequence captured in channel frame j. Thus,

$$R_{t,j} = \{r_{0,t,j}, r_{1,t,j}, r_{2,t,j}, \ldots, r_{n-2,t,j}, r_{n-1,t,j}\} \ , \tag{5.27}$$

where $r_{i,t,j} \in \mathrm{GF}(2)$.

The test responses in all channel frames are shifted sequentially into the LFSR starting with channel frame 0, while a new test is shifted in from the SRSG. The polynomials associated with the above sequences are given by:

$$R_j(x) = R_{0,j}x^{T-1} + R_{1,j}x^{T-2} + R_{2,j}x^{T-3} + \cdots + R_{T-2,j}x + R_{T-1,j} \ , \tag{5.28}$$

and

$$R_{t,j}(x) = r_{0,t,j}x^{n-1} + r_{1,t,j}x^{n-2} + r_{2,t,j}x^{n-3} + \cdots + r_{n-2,t,j}x + r_{n-1,t,j} \ . \tag{5.29}$$

Let $R_{i,j} \subset R_j$ be a sub-sequence having the same length as R_j and containing only the test responses captured at the i^{th} scan cell of channel j and zeros elsewhere. Thus,

$$R_{i,j} = \{0, \ldots, 0, r_{i,0,j}, 0, \ldots, 0, r_{i,1,j}, 0, \ldots, 0, r_{i,T-2,j}, 0, \ldots, 0, r_{i,T-1,j}, 0, \ldots\} \ , \tag{5.30}$$

and

$$\begin{aligned} R_{i,j}(x) &= r_{i,0,j}x^{n-1-i}x^{T-1} + r_{i,1,j}x^{n-1-i}x^{T-2} + \cdots + r_{i,T-2,j}x^{n-1-i}x \\ &\quad + r_{i,T-1,j}x^{n-1-i} \qquad (5.31) \\ &= x^{n-1-i}\left[r_{i,0,j}x^{T-1} + r_{i,1,j}x^{T-2} + \cdots + r_{i,T-2,j}x + r_{i,T-1,j}\right] \ . \qquad (5.32) \end{aligned}$$

Hence,

$$R_j(x) = \sum_{i=0}^{n-1} R_{i,j}(x) \ , \tag{5.33}$$

where the addition is the usual bitwise modulo-2.

Theorem 5.3.3 *The final signature S_j for any scan channel j in a GSTUMPS module, compacted by an m-bit MISR in SSC-mode, is equal to:*

$$S_j = \sum_{i=0}^{n-1} S_{i,j} \ , \tag{5.34}$$

where n is the length of the scan channel, $S_{i,j}$ is the compacted signature for the test responses captured at the i^{th} scan cell while forcing zeros in the remaining channel cells, and the MISR is reseted before any compaction.

Proof:

The final channel signature $S_j(x)$ is given by:

$$S_j(x) \overset{(a)}{=} R_j(x) \bmod G_j(x) \tag{5.35}$$

$$\overset{(b)}{=} \left[\sum_{i=0}^{n-1} R_{i,j}(x)\right] \bmod G_j(x) \tag{5.36}$$

$$\overset{(c)}{=} \sum_{i=0}^{n-1} R_{i,j}(x) \bmod G_j(x) \tag{5.37}$$

$$\overset{(d)}{=} \sum_{i=0}^{n-1} S_{i,j} \tag{5.38}$$

where,

(a) follows from Equation 2.12,

(b) follows from Equation 5.33,

(c) follows from the linearity of the XOR operation in SA, and

(d) follows from Equation 2.12 and the definition of $S_{i,j}$. Thus, the theorem is proved □.

It should be mentioned that reseting the MISR before any compaction is not a necessary condition for Theorem 5.3.3. It can be easily shown that initializing the MISR to different states, when compacting the different scan cell signatures, affects the S_j mathematics by only introducing additional bitwise-XOR operations. Thus,

$$S_j(x) = \sum_{i=0}^{n-1} S_{i,j} + k(x) \ , \tag{5.39}$$

where $k(x)$ is a modulo-2 function of the initial states used with the MISR. Initializing the MISR to the all zeros state is assumed here for the reasons mentioned in Section 5.3.1.

Corollary 5.3.2 *The total channel signature for any subset* (Ω) *of scan cells in the channel is equal to:*

$$S_{\Omega_j} = \sum_{i \in \Omega} S_{i,j} \ , \tag{5.40}$$

where the MISR is reseted before compacting any signature.

The corollary follows directly from Theorem 5.3.3. This corollary is instrumental in formulating the diagnostic algorithms presented in this thesis, as will be shown later in Chapters 6 and 7.

Chapter 6

Adaptive BIST Fault Diagnosis

The continuously increasing densities in VLSI modules with ever smaller geometries, pushes the need for BIST, design for diagnosability, and more cost-effective, time-effective and reliable BIST diagnostic techniques.

The basic objective of BIST fault diagnosis is to locate the faults in the MUT based on the compacted signatures.

Several SA-based diagnostic schemes have been developed that can be used at the system, board, MCM, and chip levels. A comprehensive survey of these techniques was presented in Chapters 3 and 4.

The basic deficiency of the majority of these techniques is their overwhelming assumption of a small number of errors in the test response sequence. This assumption is generally unrealistic since a faulty module in a practical BIST environment can generate an enormous number of erroneous bits in the TRS. The remaining techniques do not restrict the error or fault multiplicities, but suffer from a high hardware overhead and/or diagnostic complexity.

Motivated by the above, an alternative approach was investigated to circumvent the above deficiencies. A novel fault diagnosis technique is presented to be used with GSTUMPS-like scan-based BIST designs. The technique is based on multiple signature analysis.

Using a simple *adaptive* algorithm and minimal additional test hardware, the proposed approach allows for the highest diagnostic resolution combined with the ability to tradeoff the diagnosis time for the diagnostic hardware overhead.

The technique can be used at all levels of testing to provide chip, MCM, board and

system level diagnostics. The diagnosis process is *hierarchical* such that a faulty module identified at one level becomes the MUT at the next lower level.

At the system level, the technique provides PCB or MCM level diagnostics. At the PCB or MCM level, it has the ability to isolate the faulty chip(s). And finally, at the chip level, the diagnostic results can be combined with other well-developed classical diagnostic techniques to provide a gate or even transistor level diagnosis.

A scan-based BIST architecture such as GSTUMPS enables a more efficient diagnosis by providing direct access to an enormous number of internal nodes of the MUT through the scan chains.

Diagnosing various malfunctions in a module begins by identifying the faulty scan channels that capture errors during test and the *discrepant* scan cells within these channels.

Clearly, locating the faulty scan channel(s) is sufficient for system, board, and MCM level diagnostics. At the chip level, locating the discrepant scan cells is very helpful in achieving a satisfactory diagnosis.

Typically, VLSI chips are designed such that they contain *shallow* combinational logic between the successive scan cells forming the scan chains in test mode. Thus, identifying the fault capturing scan cells isolates the existing faults to a small cone of logic feeding the discrepant cells.

In addition, most of today's VLSI chips are designed at the RTL level [38], and are automatically synthesized by the design tools [56]. This means that a designer or a test engineer is more familiar with the scan cells than the inner circuits of such modules [56]. Thus, identifying the faulty cells provides a coarse localization of the failure sites. This information can then be fed to classical failure analysis tools to provide an automated gate-level or transistor-level diagnosis.

The multiplicity of faulty channels and the discrepant scan cells within a channel is not limited to any specific value. The adaptive diagnostic algorithm guarantees the correct identification of any number of them ranging from a single faulty scan channel/cell to the catastrophic case of all faulty channels/cells. The adaptive algorithm achieves this while minimizing the number of signatures that must be collected during diagnosis.

Since fault isolation is done by additional hardware guided by some basic software, implementing the adaptive algorithm, the computational space and time for fault isolation are reduced substantially.

The proposed scheme does not assume any specific fault model, and thus can diagnose all voltage-detectable faults, including delay faults [3], irrespective of their multiplicities.

In Section 6.1, the general concepts of the adaptive algorithm are described. Section 6.2 describes the details of identifying the fault-capturing scan channels and the needed diagnostic hardware. Section 6.3 extends the algorithm to locate the faulty scan cells. The performance of the adaptive algorithm is analyzed in Section 6.4. Section 6.5 reports the experimental results and validates the predicted analytical performance. The chapter concludes with Section 6.6.

6.1 Principles of the Adaptive Diagnostic Algorithm

In this section, the general concepts and principles of the adaptive algorithm are presented.

The algorithm has two modes of operation. First, in the *multiple scan channels* mode, it is employed to identify the faulty scan channel(s), i.e., the scan chain(s) designated as having error-capturing scan cell(s). Each of the identified faulty channels becomes the subject of diagnosis in the *single scan channel* mode. In this mode, the adaptive algorithm is extended to locate the discrepant scan cells within each of the identified faulty channels. The algorithm is assumed to be implemented in software.

In each mode, the complete process of diagnosis consists of a number of *Diagnostic Test Experiments* (DTEs). During each DTE, the following sequence of operations is executed:

- The GSTUMPS test core is initialized as described in Chapter 5. The RAMs and MUT sequentials are initialized, and the SRSG is loaded with its predetermined seed.
- The MUT is switched to self-test mode, a pre-determined number of pseudorandom test patterns (T) are applied to the MUT through the channels, and the test responses are shifted out to the compactor.
- At the end of the BIST session, a DONE signal stops the BIST and causes the MUT to hold its state until the final MISR signature is unloaded to the tester data files. The signature is processed by the adaptive algorithm where it is compared to its pre-computed reference.

The adaptive diagnosis is *dynamic* in the sense that a DTE is generated automatically and online during diagnosis [150]. It is also a multi-phase, *iterative* process. Rather than

generating and executing the entire set of diagnostic test experiments in a fixed order, each DTE is generated and applied upon request, i.e., it depends on the results of the *previous* DTEs and the status of the fault diagnosis process [56].

Thus, adaptive diagnosis avoids the costly steps of computing static faulty-signature dictionaries, diagnostic fault simulation, and static DTE generation. It is always *complete* with respect to the targeted faults and the signature measurements used during diagnosis.

Each DTE makes the diagnostic algorithm *well informed* about the status of the diagnosis process. The number of DTEs depends on the actual number of faulty scan channels and the number of fault-capturing scan cells within a faulty channel. The use of adaptive diagnostics may substantially reduce the *average* number of DTEs required to locate a fault.

The only difference between successive test experiments lies in their test response compaction arrangements. The test responses are gathered from different channels in MSC mode or different scan cells within a faulty channel in SSC mode. Making only a selected subset of scan elements observable by the MISR during each DTE is the essence of the adaptive algorithm.

Within the framework of the GSTUMPS architecture, a faulty scan channel is identified as follows: Signatures are compacted in DTEs corresponding to different combinations of channels. The combination of channels, made observable to the MISR by the CSC, depends on the results of signature comparison from the previous DTEs. This allows for successive test experiments to distinguish between the monitored channels and the remaining ones. Next, an off-line analysis of the compacted signatures, adaptively directs the selection of the next set of channels to be monitored, and concludes about the possible faulty scan channels. This process continues until all faulty channels are identified.

Locating the discrepant scan cells within the identified faulty channels follows exactly as above.

The off-line *reasoning* analysis used with adaptive diagnosis is based on the principle of elimination (divide and conquer). Each DTE identifies the *possibly* faulty scan elements by examining the compacted signature. If the signature produced by a set of scan elements is error-free, neglecting the controllable impact of traditional signature aliasing, then all of these channels/cells can be declared as fault-free, and can be eliminated from further analysis. This restricts the search for the discrepant scan elements in the remaining ones.

On the other hand, if the compacted signature is erroneous, then at least one of the

observed channels/cells is faulty. The diagnosis dynamically adapts to these results and directs the selection of the next group of scan elements to be monitored in the following test experiment. This continues until it is possible to deduce the locations of all the fault-capturing scan channels/cells.

As mentioned in Chapters 3 and 4, the current diagnostic schemes are inefficient because of the overhead of collecting many signatures for the fault-free channels/cells, one at a time. However in adaptive diagnosis, the scan elements are selected such that the maximum number of fault-free channels/cells can be eliminated in each DTE.

Instead of collecting signatures for every single scan channel/cell no matter whether it is faulty or not, signatures are compacted only for the channels/cells that are identified as being possibly faulty during the previous DTEs. This minimizes the average number of signatures that must be collected during diagnosis, and effectively reduces the overhead of collecting signatures for the many fault-free scan elements, while maintaining an optimal diagnostic resolution.

Clearly, the selection of the scan elements to be observed has a critical impact on the computational and hardware complexities. The selection problem can be stated as follows: Given a set of m scan channels, each having n cells, devise a sequential testing policy that unambiguously locates any set of $l \leq m$ faulty channels or $k \leq n$ discrepant scan cells within the identified faulty channels, while minimizing the average number of DTEs that must be executed.

Analytically, it can be seen that the problem is that of *searching* for l or k items in a given set of m or n elements, respectively. The best searching algorithm, in the sense that it can locate any *single* item in a given sorted set with the minimum number of *steps*, is without doubt the *binary search algorithm* [151].

The most efficient searching and sorting algorithms known to date are based on traversing *tree* structures. Tree structures are used in the adaptive algorithm to model the DTEs in the diagnosis process. Hence, a brief summary of these structures is presented next.

6.1.1 Overview of Tree Structures

A tree is a finite set of *nodes* such that each node is either the *root* or a member of a subtree. Each node can be reached via a linking pointer from a *parent node*, and it may have several *child* nodes. A tree also satisfies two additional requirements:

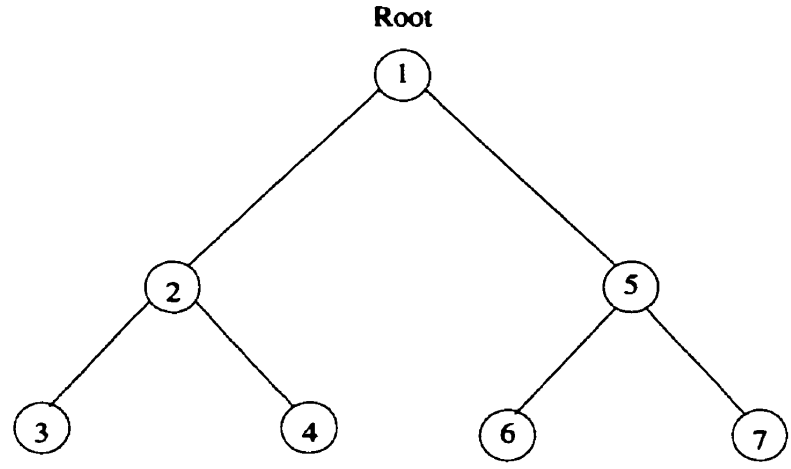


Figure 6.1: A complete binary tree structure.

- A path drawn from any node in the tree to another must be unique, if it exists at all.
- There must be exactly one node from which a path can be drawn to any other node in the tree.

A node in a tree is said to be at *level* "i" if the path to that node from the root traverses exactly i nodes. The root itself is always at level 0. The tree is said to have a height of L if it contains L levels.

The *degree* of a node is the number of child nodes attached to it. A very special case of a tree structure is the *binary tree*, in which every node is at most of degree-2, as shown in Figure 6.1.

Binary trees are called *complete* if two requirements are satisfied:

- Every level (except the last) is fully populated with nodes.
- The nodes at the last level are all placed as far to the left-hand as possible.

The number of levels in a complete binary tree having x nodes is equal to:

$$L = \min\{q | 2^q - 1 \geq x\} \ . \tag{6.1}$$

Clearly if $2^q - 1 \geq x$, then,

$$q \geq \log_2(x+1) \ , \tag{6.2}$$

and,

$$L = \lceil \log_2(x+1) \rceil \ . \tag{6.3}$$

It can be easily seen that the i^{th} level of an L-level complete binary tree must have exactly 2^i nodes, except for the last level. Correspondingly, a complete binary tree has at most $2^L - 1$ nodes and at least 2^{L-1} nodes, for all levels except the last one must be fully populated.

An L-level complete binary tree having exactly $2^L - 1$ nodes is called a *full* binary tree.

The *traversal* of a tree structure, visiting each node exactly once, is one of the most elementary and useful operations imaginable. One form of a *recursive* traversal is to:

1. visit the root node, then
2. visit all members of the left subtree, and then,
3. visit all members of the right subtree.

Ordering within the left and right subtrees is determined by the fact that subtrees are trees, so they must be traversed according to the same rule. It should be noted that a tree traversal can also be done in a reversed order: root, right subtree, and left subtree.

If a collection of items are arranged in a binary tree, it becomes a simple matter to search for a particular item. The principle is that of the classical *divide and conquer* approach, applied recursively. Searching requires traversing the tree, with a *test* at each node to determine whether it contains the item(s) sought. If the item(s) is/are found, the search terminates. Otherwise, if the traversal is completed and the object of search is not located, a search failure is reported.

6.1.2 Modeling the Adaptive Diagnostic Process

The naturally rising question is that how can the binary search algorithm be applied to the adaptive diagnostic testing ?

Well, the combinations of scan channels/cells, for which a signature must be collected in each DTE, can be thought of as being *nodes* in a binary tree.

The *root* node includes the whole set of scan channels in MSC mode or the entire set of scan cells within a faulty channel in SSC mode.

All of the scan elements at the root node are given binary addresses, and subsequently divided into two subsets according to their addresses, starting with the least significant bit at the first level of the tree. The nodes in the following levels are similarly divided into two subsets according to the address bit corresponding to that level. Thus, the terminal nodes correspond to DTEs involving a single scan channel or cell.

In order to devise a complete tree algorithm, it only remains to define what is meant by *visiting* each of the tree nodes. In the context of the current diagnostic problem, visiting a tree node means executing the DTE associated with that node.

Diagnostic experiment trees [152, 153] serve as the basis for the adaptive algorithm described in this chapter. The formal definitions of these structures are given below.

Definition 6.1.1 *A Diagnostic experiment Tree $T_D(V, E)$ is a collection of a set of Vertices $V(T_D)$ and a set of directed Edges $E(T_D)$. Each vertex $v \in V(T_D)$ of the tree is associated with a DTE. Each edge $e \in E(T_D)$ is of the form (u, v), $u, v \in V(T_D)$ with the direction of the edge being from u to v.*

The start of the diagnosis process can be represented by the *root* node $r(T_D)$ with the entire scan list being associated with it.

Definition 6.1.2 *The root of the diagnostic experiment tree $(r(T_D))$ is a unique DTE in T_D such that there is no edge of the form $(u, r(T_D))$, $\forall\ u \in V(T_D)$. The scan list associated with such a DTE is the entire list of scan channels in MSC mode, or the entire list of scan cells within a faulty channel in SSC mode.*

An edge provides the link between any *parent* DTE and its two *child* DTEs in the tree. It is associated with the outcome of the parent DTE. The set of scan elements associated with DTE_i is a subset of the one associated with its *parent* $P(\text{DTE}_i)$.

Definition 6.1.3 *The Parent of* DTE_i $(P(\text{DTE}_i))$ *is a unique DTE in T_D such that the edge $(P(\text{DTE}_i), \text{DTE}_i) \in E(T_D)$.*

Hence, all the DTEs in T_D are unique. Clearly, the root DTE (DTE_r) does not have a parent. A parent DTE and its children exist in two consecutive *levels* of the diagnostic experiment tree.

Definition 6.1.4 *The Level of* DTE_i $(L(\mathrm{DTE}_i))$ *is the length of the path between* DTE_r *and the node associated with* DTE_i *in* T_D.

6.1.3 Reference Signature Generation

The diagnostic information needed for the off-line reasoning analysis, after each test experiment, is a list of fault-free reference signatures, one for each DTE.

The reference signatures are not hardwired in the GSTUMPS module because of the extra area overhead required to store them and implement the corresponding comparators. In addition, the design of a module only becomes final at the very end of the design cycle, thus possibly creating a bottleneck if the signatures were to be hardwired.

Instead, the reference signature dictionaries are stored off-line in the diagnostic software module where they are compared with the MISR signatures unloaded after each DTE. Dictionaries are especially useful when a repeated diagnosis of different copies of the same MUT is performed [3, 154, 155].

There are two approaches to generating the reference signatures. In the first approach, the reference signatures associated with each DTE are generated separately in both MSC and SSC modes. This requires an expensive, CPU-intensive MUT simulation in BIST mode that includes the BIST circuitry.

An alternative approach is adopted that makes use of the principle of signature superposition and the analytical results developed in Section 5.3. The results of Theorem 5.3.2 and Theorem 5.3.3 allow for adding up individual signatures to construct any combination of signatures. This can be done in parallel while the MISR is compacting responses from the same combination in the MUT.

Generating reference signatures for each scan cell in the MUT is sufficient for both MSC and SSC diagnostics. In SSC mode, the reference signature for any combination of scan cells in a DTE can be simply obtained as the *bit-wise* modulo-2 addition of the corresponding reference scan cell signatures. In MSC mode, the reference channel signatures and any of their combinations can be obtained in exactly the same manner as in the SSC mode.

The reference scan-cell signatures are generated as part of the MUT design cycle by a good-machine simulation in BIST mode. When compacting a signature for any cell, zeros are shifted-in for the other cells as explained in Chapter 5.

It should be noted that only the fan-in logic cone of the scan cell under consideration needs to be kept during simulation. This reduces the simulation effort dramatically, and makes the reference signature generation process more robust. The complete signature dictionary need not be completely recomputed because of last minute engineering changes. Only the signatures of the affected scan cells need to be recomputed.

Generating the reference signatures as proposed above has a major advantage over the straightforward approach. Despite the huge amount of diagnostic data that has to be processed during the successive BIST sessions, the number of reference signatures that have to be stored is substantially smaller than the number of all possible combinations.

Hence, by incorporating the adaptive diagnostic scheme into the GSTUMPS architecture, common sources of defects can be identified in a series of BIST sessions, requiring a minimal number of reference signatures to be stored.

6.2 Identifying the Faulty Scan Channels

In this section, an adaptive algorithm is proposed that can be used with a GSTUMPS module, in MSC mode, to locate the faulty scan channels. It will be shown that the required hardware to implement this scheme comprises simple components with a minimal overhead.

As mentioned in Section 6.1, the complete process of diagnosis consists of a number of test experiments. During each DTE, the proposed algorithm decides the subset of scan channels to be made observable to the MISR. The complete set of T pseudorandom test patterns are applied to the MUT through the channels, and the test responses are shifted out to the compactor.

The straightforward approach to diagnosis is to monitor each channel separately by gating the scan-out test response data [47, 118, 132]. The MISR in this case reduces to a simple programmable LFSR. This scheme requires the repetition of the same BIST session m times, thus collecting as many signatures as the number of scan channels.

As explained in Chapter 4, this basic approach has the advantage of a very simple control hardware, the highest diagnostic resolution, and a trivial diagnostic reasoning procedure: if a channel is faulty, then its compacted signature is different from its fault-free reference.

However, in many VLSI designs the number of scan channels is made approximately equal to the number of scan cells within a channel [132]. This is done in order to reduce the test application time associated with shifting in and out the chains. Thus, the straightforward approach may only be viable for MUTs with a small number of scan channels.

To remedy these problems, the proposed algorithm systematically partitions all scan channels into clusters in an adaptive way, such that the channels observed during a DTE depend on the results of signature comparison from previous DTEs. The process is arranged in such a way that each pair of DTEs capture test responses from *orthogonal* subsets of scan channels. This minimizes the average number of signatures to be compacted, thus reducing the diagnostic test application time.

Without loss of generality, the number of scan channels (m) is assumed for the moment to be an integer power of 2. This is generalized to any m subsequently.

6.2.1 Locating the Faulty Channels

The faulty channel identification algorithm is explained first with simple examples. The complete algorithm is given in Section 6.2.2.

Consider first the case of a single faulty channel. If it is known that the fault multiplicity is $l = 1$, then the fault diagnosis problem is straightforward.

To locate the targeted faulty channel CH_i out of the m scan chains, $\{CH_0, CH_1, \ldots, CH_{m-1}\}$, the signature of the root DTE (i.e., the signature associated with all scan chains) is compacted and compared to the fault-free reference. The diagnostic experiment tree is shown in Figure 6.2

Since it was predetermined that there is a single faulty channel, then the two previous signatures will be different. Proceeding further to the next level of the tree, the channels are divided into two equal groups.

The signature of one of the latter groups is compacted and compared to its reference. If the two signatures are identical, then it can be concluded that the faulty channel is in the other group that was not tested, and that group will be the subject of diagnosis in the following DTE. However if the two signatures were different again, then the faulty channel still exists in the group of channels that were tested in the previous DTE.

In either case, the identified faulty group is further divided, and the whole procedure

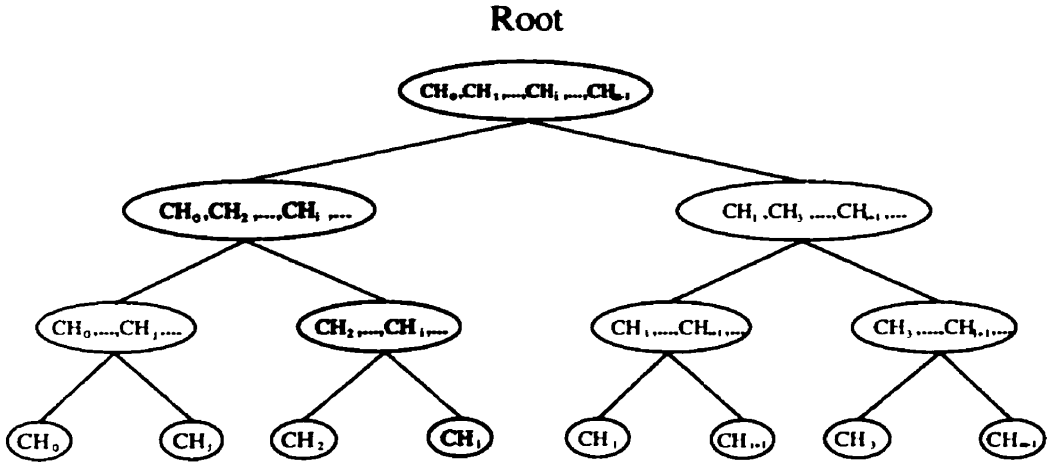


Figure 6.2: The diagnostic experiment tree T_D in MSC mode.

is repeated. The last DTE will identify the single faulty channel CH_i.

The cost of diagnosis, in terms of the number of signatures compacted, clearly depends on the number of DTEs required to locate CH_i.

Since there are $\log_2 m$ levels in T_D, not including the root node $r(T_D)$, and only one signature is compacted at each level, then the diagnostic procedure requires the compaction of $\log_2 m + 1$ signatures. This represents a tremendous saving as compared to the m signatures that were required with the straightforward approach.

However, in deriving the above procedure, it was assumed that there was only a single faulty channel. In practice, there is no way of knowing the multiplicity of the faulty channels in a MUT.

In the worst case scenario, the faulty channels can be distributed in T_D such that each of the DTEs contains a faulty chain all the way to the last level of the tree. In this case, a *minimum* of m signatures and a *maximum* of $2m - 1$ signatures must be compacted (i.e., as many nodes in T_D).

Clearly, the performance of this new diagnostic procedure as it is, is worse than the current straightforward approach. This actually was the motivation behind developing the analytical signature relations in Section 5.3.

Up to this point, no use was made of the fact that each DTE has exactly two children. Thus, if the signature of a DTE and one of its children are known, then the signature of the other child DTE can be obtained, using Equations 5.14 and 5.25, by a simple bitwise

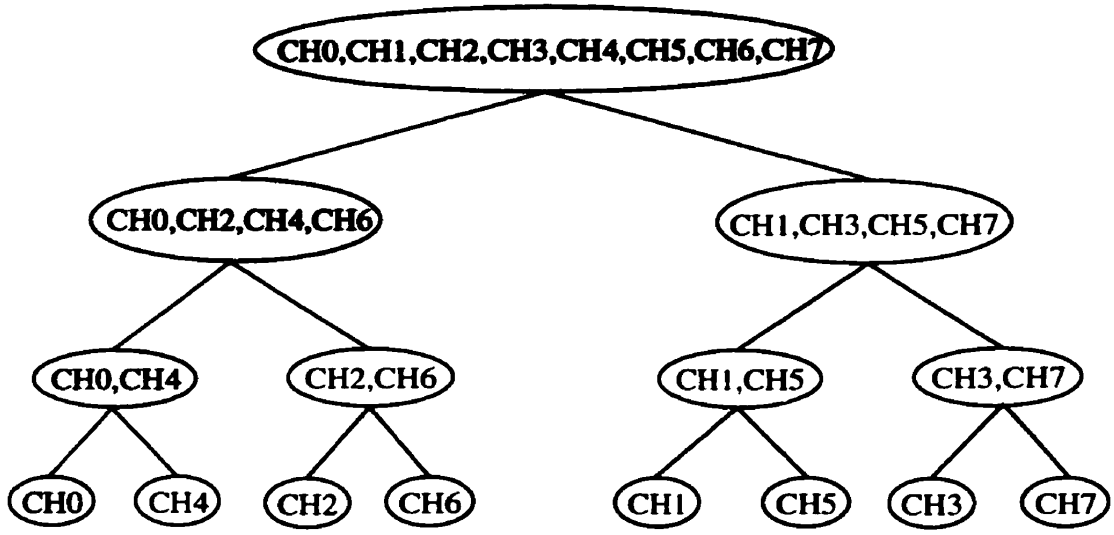


Figure 6.3: The diagnostic experiment tree for $m = 8$, $l = 1$.

XOR operations of the two signatures.

Therefore, the signatures of only half the nodes at each level of T_D must be collected in the worst case scenario, yielding a total of m DTEs in the *worst* case. Applying this modification to the adaptive diagnostic procedure, the performance in the worst case scenario is equal to the performance of the straightforward approach. All of the above can be illustrated by a simple example.

Example 6.2.1 Consider the case where the number of channels in the MUT is $m = 8$ and the fault multiplicity is $l = 1$. It should be emphasized that the fault multiplicity is not known a-priori, and it is only assumed here for the sake of explanation. In general, the diagnostic algorithm will automatically adapt according to the actual unknown number of faulty channels. The diagnostic experiment tree is shown in Figure 6.3.

Assuming that CH_0 is the fault-capturing channel, the diagnostic procedure is as follows:

1. Compact the signature of all m channels at the root DTE and compare it to its reference. Since the MUT is faulty, the root signature will differ from its fault free reference, neglecting signature aliasing.

2. Execute the DTE corresponding to the channels whose binary addresses have a "0" in the least significant bit (i.e., all the even channels), while forcing zeros in the remain-

ing channels. The compacted signature is unloaded and compared to its reference. Since CH_0 belongs to the current DTE, then the above signatures will differ.

3. The signature of the *complementary* DTE (i.e, the DTE corresponding to the odd numbered channels) is obtained by XORing the *current* signature with the previous one (i.e., the root signature). The complementary signature will not indicate the presence of any faults, and hence, the diagnostic algorithm will *adapt* and direct the test experiments to the *even* channels in the faulty DTE.

4. Since the signature of the *parent* DTE is always needed to generate the complementary one, then it must be saved off-line in a *temporary signature register*.

5. Execute the DTE corresponding to the channels whose addresses start with "00", i.e., $\{CH_0, CH_4\}$, and compare the compacted signature to its reference. The two signatures will differ again indicating the presence of a faulty channel. XORing the present signature with its parent in the temporary signature register will yield the complementary signature of the channels whose addresses start with "01", i.e.,$\{CH_2, CH_6\}$. The complementary signature will be identical to its fault-free reference.

6. Finally, the DTE corresponding to CH_0 is executed, indicating that CH_0 is faulty. The complementary signature of CH_4 is obtained again by the XOR operation and it is found to be fault-free. Hence, the faulty channel is located and the diagnostic procedure terminates.

It should be mentioned that the actual number of signatures that have been compacted is equal to $1+\log_2 m = 1+3 = 4$. Hence, for the single faulty channel case, the performance of the adaptive algorithm is of order $O(\log_2 m)$, which is the optimum performance that can be obtained with a searching algorithm in a sorted binary tree [151]. The "1" in the previous formula accounts for the root DTE.

So how does the adaptive algorithm work when multiple faulty channels are present in the MUT ? The basic procedure is identical to the one in Example 6.2.1 with few slight modifications. This is illustrated by the following example.

Example 6.2.2 Consider the case where the number of channels in the MUT is $m = 8$ and the fault multiplicity is $l = 2$. The diagnostic experiment tree is shown in Figure 6.4.

Assuming that CH_0 and CH_5 are the faulty channels, the diagnostic procedure is as follows:

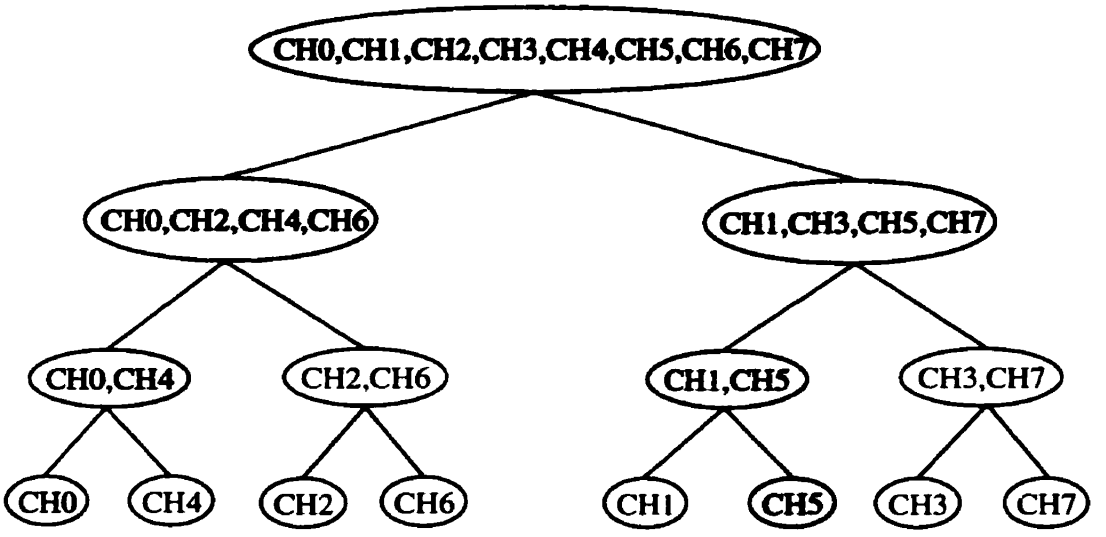


Figure 6.4: The diagnostic experiment tree for $m = 8$, $l = 2$.

1. Same as in Example 6.2.1.

2. Same as in Example 6.2.1.

3. Similar to Example 6.2.1, but in this case the XORing operation will indicate the presence of a faulty channel in the complementary DTE. Since no parallel testing is assumed here, then the diagnostic procedure continues in a sequential manner.

4. Due to the sequential nature of the algorithm, the faulty complementary signature and an identifier that indicates the channels involved in the DTE must also be saved off-line. This identifier contains the *address tag* that was used with the complementary DTE. A sequential memory structure is required to store this signature along with its identifier. This memory structure must satisfy the *last-in first-out* condition in accordance with the ordered sequential testing procedure. Hence, the required memory structure is a *signature stack*.

5. Same as in Example 6.2.1.

6. Similar to Example 6.2.1. At this point CH_0 is identified as being faulty. The test controller should recognize now that the last level of T_D has been reached, and the stack is checked for any stored signatures.

7. If the stack is empty then the diagnosis process terminates. If not, as is the case here,

the stored signature at the top of the stack is loaded into the temporary signature register and the diagnosis process continues.

8. Depending on the stored address tag, the channels are further divided into two groups. The signature of the channels whose addresses starts with "01", $\{CH_1, CH_5\}$, is compacted. The diagnosis process continues in a similar manner until CH_5 is identified as being faulty as well. The test controller recognizes again that the last level of T_D has been reached and the stack is checked again for any stored signatures. In this case, the stack is empty and the diagnosis process terminates.

The actual number of signatures that have been compacted is equal to 6. The formal adaptive algorithm is presented next.

6.2.2 The Adaptive Diagnostic Algorithm

The procedure for locating faulty channels of any multiplicity l is similar to the procedure used with the double faulty channels in Example 6.2.2. However, in order to simplify the diagnostic hardware, T_D is traversed according to the rule: root, right subtree, and left subtree. The diagnostic procedure has the following properties:

- Each DTE_i in MSC mode is assigned a binary *label* M_i between 0 and $m-1$.
- During DTE_i, only the channels with a label matching the mask M_i are observed by the MISR, while zeros are shifted in for the other channels.
- When a DTE is executed, the compacted signature is shifted out and analyzed off-line by the diagnostic algorithm. A new label mask corresponding to a new DTE is generated adaptively, and so on until all faulty channels are identified.

Using the results of Theorem 5.3.2, and Corollary 5.25 the general procedure for isolating the faulty scan channels in a GSTUMPS module can be summarized as follows:

Algorithm 6.2.1 *Adaptive Faulty Scan-Channel Identification:*

1. *Compact the signature S for all m channels in the MUT.*

2. *Generate the reference signature* S^0,

$$S^0 = \sum_{j=0}^{m-1} S_j^0 \ , \tag{6.4}$$

and compare it to S. *If* $S = S^0$, *then no faults were detected. Go to [13]. Else, there exists at least one faulty channel. Save* S *in a Temporary Signature Register (TSR) and execute the remaining steps.*

3. *Assign each channel a binary address* $A_{b-1}A_{b-2}\ldots A_0$, *where* $b = \lceil \log_2 m \rceil$. *Set* $i = 0$, *and* $M_i = 0$.

4. *Set the address tag bit* $A_i = 1$.

5. *Set the label mask* $M_i = X_{b-1}\ldots X_{i+1}A_iA_{i-1}\ldots A_0$, *where* X *denotes a don't care.*

6. *Execute* DTE_i.

7. *Calculate the complementary signature* $S_{\mathrm{DTE}_i^c}$,

$$S_{\mathrm{DTE}_i^c} = S_{TSR} + S_{\mathrm{DTE}_i} \ , \tag{6.5}$$

where DTE_i^c *is the complementary test experiment with a label mask* $M_i^c = X_{b-1}\ldots X_{i+1}\bar{A}_iA_{i-1}\ldots A_0$.

8. *Generate the complementary reference signature* $S^0_{\mathrm{DTE}_i^c}$,

$$S^0_{\mathrm{DTE}_i^c} = \sum_{j\in \mathrm{DTE}_i^c} S_j^0 \ , \tag{6.6}$$

and compare it to $S_{\mathrm{DTE}_i^c}$. *If* $S_{\mathrm{DTE}_i^c} \neq S^0_{\mathrm{DTE}_i^c}$, *then* push *it into the signature stack along with its label mask* M_i^c.

9. *Generate the reference signature* $S^0_{\mathrm{DTE}_i}$,

$$S^0_{\mathrm{DTE}_i} = \sum_{j\in \mathrm{DTE}_i} S_j^0 \ , \tag{6.7}$$

and compare it to S_{DTE_i}. *If* $S_{\mathrm{DTE}_i} = S^0_{\mathrm{DTE}_i}$, *then go to [12]. Else, save* S_{DTE_i} *in the TSR and proceed to the next step.*

10. *Set* $i = i + 1$. *If* $i < b - 1$, *go to [4], else go to [11].*

11. *Execute* DTE_{b-1}. *If* $S_{\mathrm{DTE}_{b-1}} \neq S^0_{\mathrm{DTE}_{b-1}}$, *then the scan channel* $CH_{M_{b-1}}$ *is faulty. If* $S_{\mathrm{DTE}^c_{b-1}} \neq S^0_{\mathrm{DTE}^c_{b-1}}$, *then the scan channel* $CH_{M^c_{b-1}}$ *is faulty.*

12. *Check the signature stack. If it is empty go to [13]. Else,* pop *the signature and its label mask at the top of the stack. Load the signature into the TSR, and set the current label mask to be equal to the popped one from the stack. Go to [10].*

13. *The procedure for identifying the faulty channels is complete.*

It should be mentioned that the complementary signature in step (11) is not needed if $S_{\mathrm{DTE}_{b-1}} = S^0_{\mathrm{DTE}_{b-1}}$, since its status can be implicitly deduced. However, it is calculated explicitly as an additional check to verify the validity of the diagnostic results obtained thus far.

6.2.3 Hardware Support for the Diagnostic Algorithm

The diagnostic hardware required to implement the adaptive algorithm resides in the channel selection controller (CSC), first introduced in Chapter 5. The CSC is used to gate the MISR inputs, such that only the selected subset of channel outputs, dictated by a DTE, are made observable to the compactor.

What follows is a description of a low-cost CSC, and the associated hardware components required to support the adaptive faulty-channel identification algorithm. The hardware configuration that accommodates the adaptive algorithm is shown in Figure 6.5.

The CSC consists of an m-stage Channel Select Register (CSR). Each stage is implemented with an SRS, thus forming a scan-testable shift register. Each stage is dedicated to driving one of the m 2-input AND gates used to gate the MISR inputs. The CSR is clocked with TCLK_2 in synchronization with the MISR scan-out shift clock.

The SRSs in the CSR have no system inputs. Assuming that each SRS is equivalent to 9 AND gates [1], then the hardware complexity of the CSC has order $O(10m)$.

During the normal operation of the MUT, the CSR is held to the all ones value. In the MSC diagnostic mode, the CSR is loaded with an m-bit code and a DTE is executed. The output of CSR can be thought of as an m-bit binary number representing the label associated with the subset of scan channels involved in a DTE. The CSR contents are held for the whole duration of a DTE, and new values are shifted into these SRSs when the next DTE is about to commence.

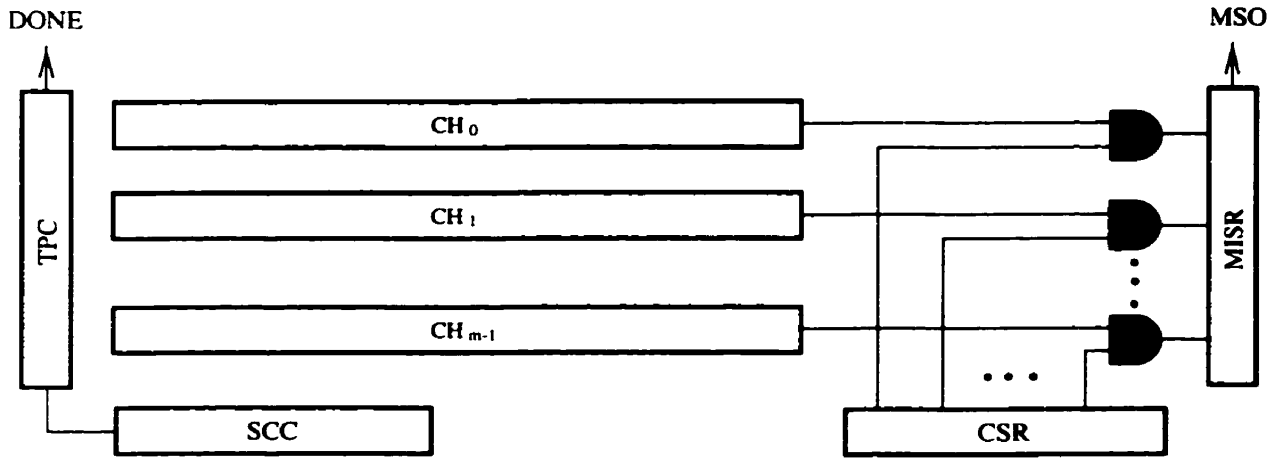


Figure 6.5: Hardware support for adaptive faulty-channel identification.

The MISR is reseted before the start of signature compaction for every DTE. The test responses from the channels are fed to the MISR in synchronization with the channels scan-out shift clocks.

In addition to the CSC, the test access port controller (TAPC), first introduced in Chapter 5, generates the signals required to control the whole diagnostic process. It serves as a front end for BIST, controlling all BIST functions. It also provides the necessary clocking signals to synchronize between the different BIST and DFT components in the MUT.

The TAPC contains the necessary hardware for manipulating the MUT interface to the external world via a standard test bus interface. It utilizes the MUT's scan paths for data transfer between the MUT, the tester and the computer hosting the software module implementing the adaptive diagnostic algorithm. The diagnosis system is explained later in Chapter 8.

During diagnosis, system control is passed to the TAPC to execute the test experiments. The TAPC supports the diagnostic algorithm through its Scan Cell Counter (SCC) and the Test Pattern Counter (TPC), two components widely employed in any scan-based BIST architecture [47]. Both counters are reseted at the beginning of each DTE, and are clocked synchronously with $TCLK_1$.

The SCC is a $(\log_2 n)$-stage binary counter, where n is the length of the longest channel. It is used to control the shifting of test responses captured in the scan cells into the MISR. For every shift, the counter is incremented. After n shifts, a single pseudorandom test is

completed and a carry-out signal is generated at the output of the counter. The SCC is reseted due to the decoded carry-out signal which in turn enables the TPC to increment. The process continues until all pseudorandom tests in a BIST session are applied.

The TPC is a $(\log_2 T)$-stage binary counter, where T is the number of pseudorandom test patterns in a BIST session. After T tests, a carry-out signal occurs at the output of the TPC. This DONE signal is used to flag the completion of a DTE.

Consequently, the signature compacted in the MISR is shifted-out through the MSO output and saved for off-line analysis in the software module. The off-line processing by the diagnostic module is used to decide the next DTE to be executed. The whole process is repeated until an adequate number of signatures is collected for a complete diagnosis.

6.3 Identifying the Faulty Scan Cells

In this section, an adaptive diagnostic algorithm is presented that can be used with a GSTUMPS module in SSC mode, to locate the discrepant scan cells. It will be shown that hardware required to implement this scheme comprises simple components with a minimal overhead.

The technique presented in Section 6.2 for the faulty-channel identification can also be extended to locate the fault-capturing scan cells within the identified faulty channel(s).

The diagnosis process is divided into two stages. In the first stage, the adaptive algorithm is employed in MSC mode to identify the faulty scan channels. During the second stage, each of the identified faulty channels becomes the subject of the same diagnostic procedure, now however, performed in SSC mode with respect to the scan cells within a channel.

The second stage is repeated for each of the identified faulty channels, one chain at a time. This is done by gating the MISR inputs such that only the channel under diagnosis is made observable to the MISR while zeros are forced at the other inputs. Thus, the MISR in this case reduces to an equivalent LFSR.

After each DTE in SSC mode, the off-line analysis is used to decide which scan cells are fault-free and which are possibly faulty. For the latter ones, new DTEs are adaptively generated, and the diagnosis continues until all the fault-capturing scan cells are identified.

As the number of scan channels can often be much smaller than the number of scan

cells within a channel [38], the diagnosis time in the first stage can be significantly shorter than the second stage.

Without loss of generality, it will be assumed that the number of scan cells in each channel is n, where n is the number of scan cells in the longest channel.

6.3.1 The Adaptive Diagnostic Algorithm

The procedure for locating the discrepant scan cells is identical to the procedure used for identifying the faulty scan channels described in Section 6.2.2.

The diagnostic procedure has the following properties:

- In order to locate the discrepant scan cells, the identified faulty channels are diagnosed one at a time.
- Each DTE_i in SSC mode is assigned a binary *label* N_i between 0 and $n-1$.
- During DTE_i, only the scan cells with a label matching the mask N_i are observed by the MISR, while zeros are shifted-in for the other cells.
- When a DTE is executed, the compacted signature is shifted out and analyzed off-line by the diagnostic algorithm. A new label mask corresponding to a new DTE is generated adaptively, and so on until all the discrepant cells are identified. The procedure then repeats for the remaining faulty scan channels.

Using the results of Theorem 5.3.2, Theorem 5.3.3, Corollary 5.25, and Corollary 5.3.2, the procedure for isolating the fault-capturing scan cells in a GSTUMPS module can be summarized as follows:

Algorithm 6.3.1 *Adaptive Faulty Scan-Cell Identification:*

1. *Execute Algorithm 6.2.1 and identify the faulty scan channels. Maintain their signatures off-line for further analysis.*
2. *For each of the identified faulty channels, execute the remaining steps. If no faults were detected, go to [15].*

3. *Obtain the signature $S(x)$ for all n scan cells in the faulty channel (j) under diagnosis, and save it in a TSR.*

4. *Assign each cell a binary address $A_{b-1}A_{b-2}\ldots A_0$, where $b = \lceil \log_2 n \rceil$. Set $i = 0$, and $N_i = 0$.*

5. *Set the address tag bit $A_i = 1$.*

6. *Set the label mask $N_i = X_{b-1}\ldots X_{i+1}A_iA_{i-1}\ldots A_0$, where X denotes a don't care.*

7. *Execute* DTE_i.

8. *Calculate the complementary signature $S_{\mathrm{DTE}_i^c}$,*

$$S_{\mathrm{DTE}_i^c} = S_{TSR} + S_{\mathrm{DTE}_i} \ , \tag{6.8}$$

where DTE_i^c *is the complementary test experiment with a label mask $N_i^c = X_{b-1}\ldots X_{i+1}\bar{A}_iA_{i-1}\ldots A_0$.*

9. *Generate the complementary reference signature $S^0_{\mathrm{DTE}_i^c}(x)$,*

$$S^0_{\mathrm{DTE}_i^c}(x) = \sum_{k\in \mathrm{DTE}_i^c} S^0_{k,j}(x) \ , \tag{6.9}$$

and compare it to $S_{\mathrm{DTE}_i^c}(x)$. If $S_{\mathrm{DTE}_i^c}(x) \neq S^0_{\mathrm{DTE}_i^c}(x)$, then push *it into a signature* stack *along with its label mask N_i^c.*

10. *Generate the reference signature $S^0_{\mathrm{DTE}_i}(x)$,*

$$S^0_{\mathrm{DTE}_i}(x) = \sum_{k\in \mathrm{DTE}_i} S^0_{k,j}(x) \ , \tag{6.10}$$

and compare it to $S_{\mathrm{DTE}_i}(x)$. If $S_{\mathrm{DTE}_i}(x) = S^0_{\mathrm{DTE}_i}(x)$, then go to [13]. Else, save S_{DTE_i} in the TSR and proceed to the next step.

11. *Set $i = i + 1$. If $i < b - 1$, go to [5], else go to [12].*

12. *Execute* DTE_{b-1}. *If $S_{\mathrm{DTE}_{b-1}}(x) \neq S^0_{\mathrm{DTE}_{b-1}}(x)$, then the scan cell $SC_{N_{b-1}}$ is faulty. If $S_{\mathrm{DTE}^c_{b-1}}(x) \neq S^0_{\mathrm{DTE}^c_{b-1}}(x)$, then the scan cell $SC_{N^c_{b-1}}$ is faulty.*

13. *Check the signature stack. If it is empty go to [14]. Else,* pop *the signature and the label mask at the top of the stack. Load the signature into the TSR, and set the current label mask to be equal to the popped one from the stack. Go to [11].*

14. If there are remaining faulty channels to be diagnosed, go to [3]. Else, go to [15].

15. The procedure for identifying the faulty scan cells is complete.

It should be mentioned that the complementary signature in step [12] is not needed if $S_{\mathrm{DTE}_{b-1}}(x) = S^0_{\mathrm{DTE}_{b-1}}(x)$, since its status can be implicitly deduced. However, it is calculated explicitly as an additional check to verify the validity of the diagnostic results obtained thus far.

6.3.2 Hardware Support for the Diagnostic Algorithm

What follows is a description of a low-cost CSC, and the associated hardware required to support the adaptive faulty scan-cell identification algorithm.

The required diagnostic hardware resides in the channel selection controller, described in Section 6.2.3. The CSC is used to gate the MISR inputs, such that only the selected subset of scan channels/cells, dictated by a DTE, are made observable to the compactor. The hardware configuration that accommodates the algorithm is shown in Figure 6.6.

The CSC comprises an m-stage CSR, a $(\log_2 n)$-stage Scan Cell Select Register (SCSR), and a $(\log_2 n)$-stage Scan Cell Mask Register (SCMR). In addition, the CSC contains some comparing and masking logic.

Each stage of the above registers is implemented with a SRS, thus forming a scan-testable shift register. The SRSs have no system inputs.

Each CSR stage is dedicated to driving one of the m 2-input AND gates used to gate the MISR inputs. The CSR is clocked with TCLK_2 in synchronization with the MISR scan-out shift clock.

The SCSR, SCMR and the compare and mask logic associated with them are used in conjunction with the SCC to gate the MISR inputs during the faulty scan-cell identification. The SCSR and SCMR are also clocked with TCLK_2 in synchronization with the MISR scan-out clock.

During the normal operation of the MUT, the CSR is held to the all ones value while the MSC/SSC line is held high. During diagnostic testing, the circuit in Figure 6.6 has two modes of operation when executing the DTEs.

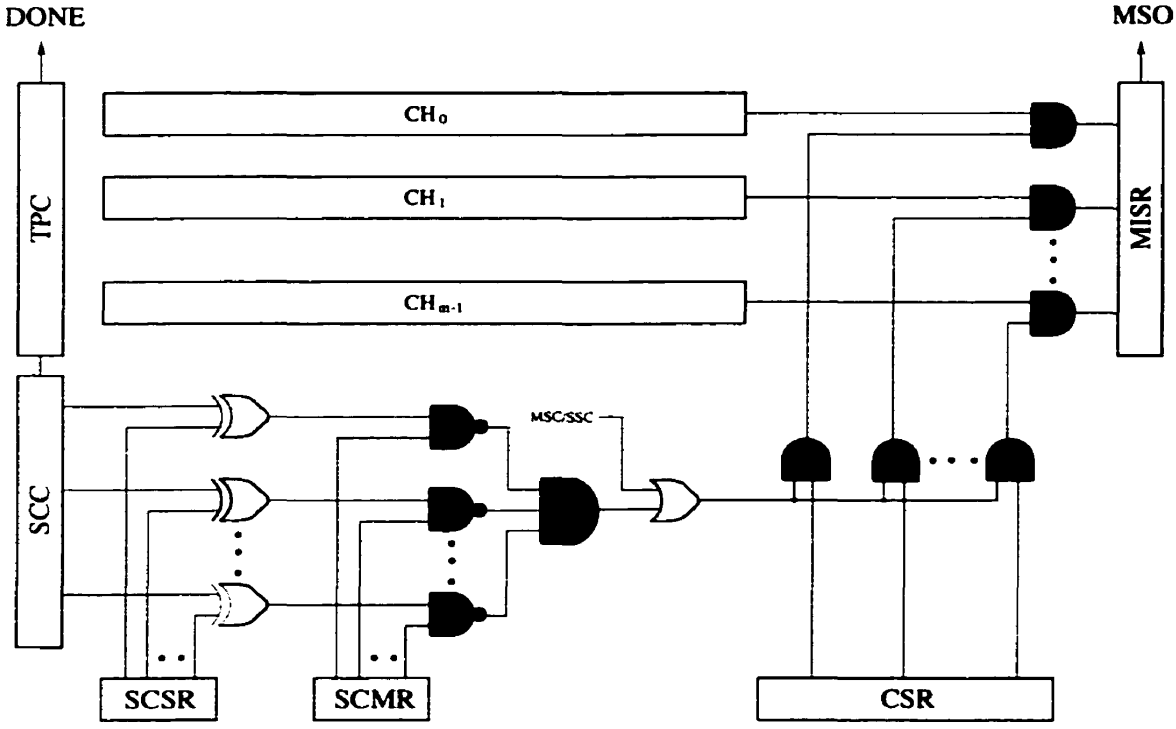


Figure 6.6: Hardware support for adaptive faulty scan-cell identification.

In the MSC diagnostic mode, the MSC/SSC is held high, the CSR is loaded with an m-bit code and a DTE is executed. The output of CSR can be thought of as the m-bit binary number representing the label associated with the subset of scan channels involved in a DTE. The CSR contents are held for the whole duration of a DTE, and new values are loaded into these SRSs when the next DTE is about to commence.

The MISR is reseted before the start of signature compaction for every DTE. The test responses from the channels are fed to the MISR in synchronization with the channels scan-out shift clocks.

For each of the identified faulty channels, the second stage of diagnosis is executed. In the SSC diagnostic mode, the MSC/SSC line is held low, and the CSR is loaded with a 1-out-of-m code word corresponding to the faulty channel being diagnosed. Test results from only that channel are fed to the MISR.

The SCSR and SCMR are loaded with $(\log_2 n)$-bit codes, and the diagnostic procedure described in Section 6.3.1 is executed to identify the error-capturing scan cells.

The output of the SCSR can be thought of as a binary number representing the label

associated with the subset of scan cells involved in a DTE. The SCMR masks-out the don't care bits in the SCSR label. The contents of SCSR and SCMR are held for the whole duration of a DTE, and new values are loaded into their SRSs when the next DTE is about to commence.

With every scan shift clock, the output of the SCSR is compared with the current scan-cell index produced by the SCC. If a match occurs, the corresponding bit from the scan channel enters the MISR.

The TAPC supports the diagnostic process, through its SCC and TPC, in the same manner described in Section 6.2.3. Both SSC and TPC are reseted at the beginning of each DTE, and are clocked synchronously with TCLK_1.

6.4 Performance Analysis of the Adaptive Algorithm

The effectiveness of the adaptive algorithm is measured by the time it takes to identify all faulty scan channels or scan cells within a faulty channel.

Let P denote the *time complexity* of applying the adaptive algorithm to a diagnostic experiment tree T_D. Also, let N_{DTE} be the number of test experiments required to locate all faulty scan channels or scan cells within a faulty channel.

Each DTE has a constant overhead (Δ_{DTE}) associated with the time it takes to execute the BIST session, unload the compacted signature to the tester for off-line analysis, and adaptively initiate the next DTE, if the diagnosis is not complete.

Thus, the time complexity of the adaptive algorithm is equal to:

$$P = \sum_{i=1}^{N_{\text{DTE}}} \Delta_{\text{DTE}_i} \ . \tag{6.11}$$

Clearly, the time associated with executing a BIST session is constant. Since the amount of time required to perform the off-line analysis is negligible compared to the amount of time required to execute a DTE, then Δ_{DTE_i} can be assumed to be a constant. The *average* diagnostic test time can thus be formulated as :

$$P_{avg} = N_{\text{DTE}} \Delta_{\text{DTE}} \ . \tag{6.12}$$

Hence, the performance of the adaptive algorithm can be measured in terms of the number of executed diagnostic test experiments.

In the following, an analytical performance model is developed that relates the number of DTEs to the number of faulty scan channels or discrepant scan cells within a faulty channel. An upper bound and a lower bound on N_{DTE} are developed, as well as the optimum N_{DTE} as predicted by Information Theory [156].

6.4.1 Deriving An Upper Bound on the Number of DTEs

To estimate the required time for diagnosis, it is assumed that T_D is fully populated or nearly so. If it is fully populated, it contains $L = 1 + \log_2 x$ levels, where x denotes the number of scan channels or scan cells within a channel. If T_D is nearly complete (perhaps lacking some terminal nodes at the last level), it must have $L = 1 + \lceil \log_2 x \rceil$ levels.

As mentioned in Section 6.1.1, there are 2^i nodes at the i^{th} level of T_D except for the last level. Since only half of the DTEs are actually executed at each level, then

$$N_{\mathrm{DTE}}(i) = \frac{1}{2}2^i \ , \tag{6.13}$$

except for $r(T_D)$, where a single DTE is executed.

Since the total number of DTEs in T_D (including DTE_r) is $2x - 1$, then the number of DTEs in the last level (N_{DTE_L}) is such that:

$$\begin{aligned} 2x - 1 &= \sum_{i=0}^{\lceil \log_2 x \rceil - 1} 2^i + N_{\mathrm{DTE}_L} && (6.14) \\ &= \frac{1 - 2^{\lceil \log_2 x \rceil}}{1 - 2} + N_{\mathrm{DTE}_L} && (6.15) \\ &= 2^{\lceil \log_2 x \rceil} - 1 + N_{\mathrm{DTE}_L} \ . && (6.16) \end{aligned}$$

Hence,

$$N_{\mathrm{DTE}_L} = 2x - 2^{\lceil \log_2 x \rceil} \ . \tag{6.17}$$

Assuming that the fault multiplicity of the scan elements is f, the required number of DTEs in the diagnostic procedure can be calculated as follows:

If $f \geq \frac{x}{2}$, then clearly, $N_{\mathrm{DTE}} = x$. On the other hand, if $f < \frac{x}{2}$, then the calculations are more involved. Since the worst case performance is considered here, it is assumed that the faulty scan elements are distributed such that all f faults at the terminal nodes of T_D

come form different parent nodes in the preceding levels. Since the last level of T_D may not be fully populated, then two cases must be examined.

First, if $f > \frac{N_{\mathrm{DTE}_L}}{2}$, then,

$$N_{\mathrm{DTE}}(f) = 1 + \frac{1}{2}\sum_{i=1}^{\lceil \log_2 f \rceil} 2^i + \sum_{i=\lceil \log_2 f \rceil + 1}^{\lceil \log_2 x \rceil - 1} f + \frac{N_{\mathrm{DTE}_L}}{2} \tag{6.18}$$

$$= 1 + \frac{1}{2}\left[\sum_{i=0}^{\lceil \log_2 f \rceil} 2^i - 1\right] + f\left(\lceil \log_2 x \rceil - \lceil \log_2 f \rceil - 1\right) \tag{6.19}$$
$$+(x - 2^{\lceil \log_2 x \rceil - 1})$$

$$= \frac{1}{2} + \frac{1}{2}\left[\frac{1 - 2^{\lceil \log_2 f \rceil + 1}}{1 - 2}\right] + f\left(\lceil \log_2 x \rceil - \lceil \log_2 f \rceil - 1\right) \tag{6.20}$$
$$+(x - 2^{\lceil \log_2 x \rceil - 1})$$

$$= 2^{\lceil \log_2 f \rceil} + f\left(\lceil \log_2 x \rceil - \lceil \log_2 f \rceil - 1\right) + x - 2^{\lceil \log_2 x \rceil - 1} \ . \tag{6.21}$$

Second, if $f \leq \frac{N_{\mathrm{DTE}_L}}{2}$, then,

$$N_{\mathrm{DTE}}(f) = 1 + \frac{1}{2}\sum_{i=1}^{\lceil \log_2 f \rceil} 2^i + \sum_{i=\lceil \log_2 f \rceil + 1}^{\lceil \log_2 x \rceil} f \tag{6.22}$$

$$= 2^{\lceil \log_2 f \rceil} + f\left(\lceil \log_2 x \rceil - \lceil \log_2 f \rceil\right) \ . \tag{6.23}$$

Hence, the total number of DTEs that will be executed during diagnosis is bounded by:

$$N_{\mathrm{DTE}_{\max}}(f) \leq \begin{cases} 1 & : \ f = 0 \\ 2^{\lceil \log_2 f \rceil} + f\left(\lceil \log_2 x \rceil - \lceil \log_2 f \rceil\right) & : \ 1 \leq f \leq \frac{N_{\mathrm{DTE}_L}}{2} \\ 2^{\lceil \log_2 f \rceil} + f\left(\lceil \log_2 x \rceil - \lceil \log_2 f \rceil - 1\right) + x - 2^{\lceil \log_2 x \rceil - 1} & : \ \frac{N_{\mathrm{DTE}_L}}{2} < f < \frac{x}{2} \\ x & : \ f \geq \frac{x}{2} \end{cases} \tag{6.24}$$

The maximum number of complementary DTEs is of course equal to:

$$N_{\mathrm{DTE}^c_{\max}}(f) = N_{\mathrm{DTE}_{\max}}(f) - 1 \ . \tag{6.25}$$

It should be emphasized that the above formulas represent an upper limit or a worse case scenario on the number of test experiments that must be executed. Depending on the *spatial* distribution of the faulty channels/cells with respect to each other in T_D, the actual number of DTEs may be well below this upper limit for any fault multiplicity $f < \frac{x}{2}$.

6.4.2 Deriving a Lower Bound on the Number of DTEs

A lower bound on the number of test experiments can be similarly calculated. Since the best case performance is considered here, it is assumed that the faulty scan elements are distributed such that each pair of faults at the terminal nodes of T_D come form the same parent node in the preceding levels.

The minimum number of DTEs can be computed as follows. If $f \leq \frac{N_{\mathrm{DTE}_L}}{2}$, then starting at the last level in T_D, the following number of DTEs must be executed:

$$N_{\mathrm{DTE}}(f) = 1 + \sum_{i=1}^{\lceil \log_2 x \rceil} \left\lceil \frac{f}{2^i} \right\rceil \ , \tag{6.26}$$

where the "1" in the above formula corresponds to DTE_r.

Otherwise, if $f > \frac{N_{\mathrm{DTE}_L}}{2}$, then,

$$N_{\mathrm{DTE}}(f) = 1 + \frac{N_{\mathrm{DTE}_L}}{2} + \sum_{i=1}^{\lceil \log_2 x \rceil - 1} \left\lceil \frac{f - \frac{N_{\mathrm{DTE}_L}}{2}}{2^i} \right\rceil \tag{6.27}$$

$$= 1 + x - 2^{\lceil \log_2 x \rceil - 1} + \sum_{i=1}^{\lceil \log_2 x \rceil - 1} \left\lceil \frac{f - x + 2^{\lceil \log_2 x \rceil - 1}}{2^i} \right\rceil \ . \tag{6.28}$$

Hence, the minimum number of DTEs that will be executed during diagnosis is bounded by:

$$N_{\mathrm{DTE}_{\min}}(f) \geq \begin{cases} 1 + \sum_{i=1}^{\lceil \log_2 x \rceil} \left\lceil \frac{f}{2^i} \right\rceil & : \ f \leq \frac{N_{\mathrm{DTE}_L}}{2} \\ 1 + x - 2^{\lceil \log_2 x \rceil - 1} + \sum_{i=1}^{\lceil \log_2 x \rceil - 1} \left\lceil \frac{f - x + 2^{\lceil \log_2 x \rceil - 1}}{2^i} \right\rceil & : \ f > \frac{N_{\mathrm{DTE}_L}}{2} \end{cases} \tag{6.29}$$

The minimum number of complementary DTEs is of course equal to:

$$N_{\mathrm{DTE}^c_{\min}}(f) = N_{\mathrm{DTE}_{\min}}(f) - 1 \ . \tag{6.30}$$

It should be emphasized that the above formulas represent a lower limit or a best case scenario on the number of test experiments that must be executed. Depending on the spatial distribution of the faulty channels/cells with respect to each other in T_D, the actual number of DTEs may be well above this limit.

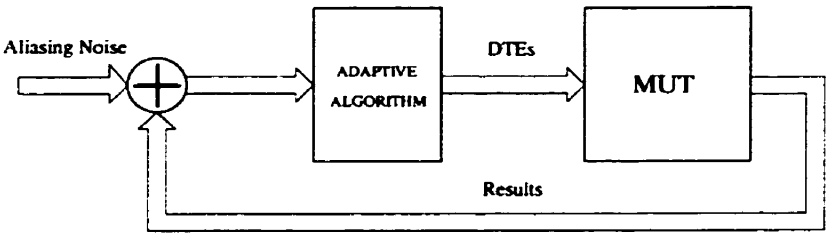


Figure 6.7: The coding theory representation of the adaptive diagnostic process.

6.4.3 Analogy Between Adaptive Diagnosis and Coding Theory

In coding theory [84, 156], the objective is to transmit messages over a noisy communication channel efficiently. Each message is coded as a sequence of *digits*. The objective is to devise a coding algorithm that results in *code words* with the smallest number of digits.

The coding theory representation of the adaptive diagnostic process is shown in Figure 6.7. Let f be the fault multiplicity which takes on values in $\mathcal{F} = \{f_0, f_1, \ldots, f_x\}$. The degenerate multiplicity f_0 indicates that the MUT is fault-free.

In the context of diagnosis, the MUT can be modeled as an information source that produces an *event* (a set of f_i-tuple faults), the outcome of which $z_j = \{x_0, x_1, \ldots, x_{f_i-1}\}$ is selected at *random* according to a probabilistic distribution.

The set $\{z_j\}_{j=0}^{J}$ are the possible outcomes of the source. This set is called the *alphabet* of the source, and since it is finite, the MUT is a finite-alphabet source.

In general, it is known from the mathematics of combinations [157, 158] that there are $x!/[(x - f_i)!f_i!]$ distinct ways of grouping f_i elements out of x distinct ones. This is the number of different possible combinations of f_i-tuple faults that can be chosen from a set of x channels/scan-cells. Thus, the MUT produces an alphabet of size J, where,

$$J = \begin{pmatrix} x \\ f_i \end{pmatrix} . \tag{6.31}$$

The MUT takes on one of the outcomes $\{z_0, z_1, z_2, \ldots, z_{J-1}\}$, with probability $p(z_j)$.

The *binary* (faulty/fault-free) signature results deduced from the DTEs can be viewed as *bits* in a *signature syndrome word*. The set of applied DTEs is *sufficient* in the sense that it can identify all possible faults in the MUT, and the DTEs are assumed to be *noiseless* in the sense that their outcomes are not corrupted by traditional signature aliasing.

The analogy between the adaptive testing problem and message coding is as follows [159, 160]: The faulty scan channels/cells in the MUT correspond to the binary messages to be transmitted, the adaptive algorithm is similar to the coding scheme, the sequence of DTE results identifying the faults is similar to the message code word, and the average length of the code word corresponds to the average number of DTEs.

The diagnostic experiment tree T_D can be regarded as a *prefix* code [84, 156] in the sense that if the address of a fault (corresponding to a terminal node in T_D) is a codeword, then no codeword is the prefix of another codeword.

The only difference between the two is that the adaptive testing problem is constrained by the available DTEs, while the message coding problem has no constraints.

Without loss of generality, assume that each DTE has unit cost. Then the expected cost of T_D is the same as the expected codeword length. Therefore finding a test tree with the minimum expected cost is the same as finding a prefix code with the minimum expected length [152].

The solution to the above coding problem is given by Huffman [84, 156]. The expected Huffman codeword length is given by the entropy (H), where,

$$H = \sum_j p(z_j) \log_2 \frac{1}{p(z_j)} \ . \tag{6.32}$$

Assuming that all J outcomes are equally likely, then the entropy of the MUT is equal to:

$$\begin{aligned} H &= \sum_{j=0}^{J-1} p(z_j) \log_2 \frac{1}{p(z_j)} && (6.33) \\ &= J\left(\frac{1}{J}\right) \log_2 J && (6.34) \\ &= \log_2 J && (6.35) \\ &= \log_2 \binom{x}{f_i} \ \text{DTEs/diagnosis} \ . && (6.36) \end{aligned}$$

The average Huffman codeword length provides a lower bound on the average length of any test sequencing algorithm including the optimal one [159, 160].

Therefore, it can be concluded using coding theory, that the minimum number of DTEs (*syndrome bits*) required to fully diagnose all possible faulty scan elements of any multiplicity f must satisfy:

$$H \leq N_{\mathrm{DTE}}(f) + N_{\mathrm{DTE}^c}(f) \tag{6.37}$$

$$< 2N_{\mathrm{DTE}}(f) \ . \tag{6.38}$$

Hence,

$$N_{\mathrm{DTE}}(f) > \frac{1}{2}\left\lceil \log_2 \binom{x}{f} \right\rceil \ . \tag{6.39}$$

6.5 Experimental Results

Evaluating the feasibility of the proposed algorithm, requires estimating the time required to diagnose a module. The diagnosis time is proportional to the number of DTEs that must be executed to locate the fault-capturing scan channels/cells. To this end, several simulation experiments were conducted using different numbers of scan elements and various fault multiplicities. This section reports the experimental results.

These experiments were also used to validate the analytical performance bounds, developed in Section 6.4.

In order to analyze the effectiveness of the adaptive algorithm, it is necessary to define a metric of the *Diagnostic Efficiency* (DE).

The efficiency of the adaptive scheme is a function of the ratio of executed DTEs to the number of scan elements. Analytically,

$$DE = \left(1 - \frac{N_{\mathrm{DTE}}}{x}\right) \times 100\% \ , \tag{6.40}$$

where x is the number of scan channels or scan cells within a faulty channel.

This metric reflects more accurately the savings in the diagnostic effort provided by the adaptive algorithm as compared to the bottom-line solution for which the number of DTEs is always equal to x.

6.5.1 Experimental Setup

For an exact performance evaluation of the adaptive algorithm, all possible combinations of faults of multiplicity f that can be chosen from a set of x scan channels/cells, must be simulated.

There are many problems with this approach. First of all, for practical scan sizes and fault multiplicities, the number of possible combinations is exponential in nature and can explode easily for even relatively small x and f. The number of such combinations is equal to:

$$J = \begin{pmatrix} x \\ f \end{pmatrix} \quad . \tag{6.41}$$

Second, due to the dimension of the problem, the run time of the simulations increases exponentially, from a few days to weeks on a Sun SPARC-20 machine, and requires enormous CPU resources.

To cope with this complexity, the possibility of computing approximate diagnostic metrics was explored. The approximate measures use *random fault sampling*.

In random fault sampling, the complexity of an exhaustive fault simulation is contained to a *randomly* selected, small subset of all possible combinations of f-tuple faults that can be chosen from a set of x scan elements.

Let j be the number of randomly generated faults for a given multiplicity f_i. Then, for each simulation experiment, the computed diagnostic metrics correspond to an average value taken over the simulated f_i-tuple faults.

Thus, the average number of DTEs is approximately equal to:

$$\overline{N}_{\mathrm{DTE}} \approx \frac{\sum_j N_{\mathrm{DTE},j}}{j} \quad , \tag{6.42}$$

and the average diagnostic efficiency is given by:

$$\overline{DE} = \left(1 - \frac{\overline{N}_{\mathrm{DTE}}}{x}\right) \times 100\% \quad . \tag{6.43}$$

It can be shown using probability theory [122] that $\|N_{\mathrm{DTE}} - \overline{N}_{\mathrm{DTE}}\|$ is very small with a very high probability. Hence, random simulations can be used as an effective way of measuring the performance of the adaptive algorithm.

The adaptive algorithm was simulated with a C program. The program reads the number of scan elements (x), the maximum fault multiplicity (f), and the number of faults to be randomly generated and evaluated (j).

For each fault multiplicity, the erroneous scan elements were chosen by having a generator of uniformly distributed pseudorandom numbers between 0 and $x-1$ invoked as many times as required to obtain an f_i-tuple with distinct entries.

The random generator used was the Unix© *erand48*(), which returns a pseudorandom value between 0 and 1. This value is multiplied by x and then rounded off to the closest integer to obtain the index of the erroneous scan element.

The experiments were repeated for different number of scan elements. For simplicity and exactness of results, the values of x were chosen to be an integer power of 2.

The program calculates the average number of DTEs required to diagnose each multiplicity of faults f_i up to f. It also reports the minimum number ($N_{\mathrm{DTE}_{\mathrm{min}}}$) and the maximum number ($N_{\mathrm{DTE}_{\mathrm{max}}}$) of test experiments for each multiplicity f_i. The effect of signature aliasing was neglected in these experiments.

The entropy of T_D is also reported as a measure of the diagnostic complexity, and a theoretical lower bound on the average number of DTEs.

Finally, the theoretical bounds on the number of DTEs ($N^T_{\mathrm{DTE}_{\mathrm{max}}}$ and $N^T_{\mathrm{DTE}_{\mathrm{min}}}$), developed in Section 6.4, are also evaluated for the same data.

All of the reported numbers do not include the extra diagnostic test experiment DTE_r.

Since x is chosen to be an integer power of 2, the average number of complementary DTEs ($\overline{N}_{\mathrm{DTE}^c}$) is equal to $\overline{N}_{\mathrm{DTE}}$, and so it is not explicitly evaluated.

It should be mentioned that the simulation program does not build T_D explicitly. The simulation complexity and memory requirements of such approach is enormous, making the simulations only practical for small x. Instead, a fault look-up table is used, which lists the relative distances of the faulty scan elements to element 0. Finding the required number of DTEs then reduces to adding the relative distances of the erroneous channels/cells within a generated f_i-tuple fault.

The simulation program can be summarized by the following pseudo-code:

Procedure 6.5.1 *Performance Evaluation of the Adaptive Algorithm:*

1. *Read x, f, and j*
2. *Create a fault look-up table.*

3. *Calculate the relative distances in the fault table.*

4. *For all fault multiplicities f_i up to f:*

   (a) *Calculate the entropy H of T_D.*

   (b) *Calculate the theoretical bounds $N^T_{\mathrm{DTE}_{\max}}$ and $N^T_{\mathrm{DTE}_{\min}}$.*

   (c) *For all j faults to be simulated:*

      i. *Generate a random f_i-tuple fault.*

      ii. *Calculate $N_{\mathrm{DTE},j}$ using the fault table.*

      iii. *Report $N_{\mathrm{DTE}_{\max},j}$ and $N_{\mathrm{DTE}_{\min},j}$.*

   (d) *Calculate the average number of DTEs:*

$$\overline{N}_{\mathrm{DTE}} \approx \frac{\sum_j N_{\mathrm{DTE},j}}{j} \ . \tag{6.44}$$

   (e) *Calculate the average diagnostic efficiency:*

$$\overline{DE} = \left(1 - \frac{\overline{N}_{\mathrm{DTE}}}{x}\right) \times 100\% \ . \tag{6.45}$$

6.5.2 Choosing the Simulation Variables

The number of scan elements simulated in the experiments were chosen between 128 and 8192. These sizes are practical and fit well with the sizes reported in the literature. For example in [38], the number of scan channels varied from 1 to 13. The number of scan cells within a channel varied from a minimum of 213, to a maximum of 6165 in 13 chains. In [161], the length of the chain used was 8000 bits.

The fault multiplicities in our experiments were bounded by $1 \leq f \leq 1024$. These numbers where chosen based on the fact that for most circuit nodes, a defect can only affect a limited number of scan cells [38]. However, global circuit nodes, such as reset or clock, can still affect a large number of scan cells. The upper bound on f can be easily determined by analyzing the structure of the MUT.

The experimental results presented in [38] show that a small f can usually achieve a reasonable high coverage. A good example is the s38417 of the ISCAS89 benchmark circuits, which contains 2048 scan cells. In this circuit, 99.54% of the circuit nodes can

f	H	$\overline{N_{\mathrm{DTE}}}$	$N_{\mathrm{DTE}_{\mathrm{max}}}$	$N^T_{\mathrm{DTE}_{\mathrm{max}}}$	$N_{\mathrm{DTE}_{\mathrm{min}}}$	$N^T_{\mathrm{DTE}_{\mathrm{min}}}$	$\%\overline{DE}$
1	7.0000	7.0000	7	7	7	7	94.5312
2	12.9886	12.0550	13	13	7	7	90.5820
4	23.3467	20.4718	23	23	8	8	84.0064
8	40.3788	33.6979	39	39	17	11	73.6735
16	66.3391	53.2569	63	63	35	18	58.3930
32	100.2213	79.5924	92	95	61	33	37.8184
64	124.1714	109.1712	121	127	94	64	14.7100

Table 6.1: Simulation results for $x = 128$, and $j = 100000$.

affect at most 10 cells if any of them fails. These results were obtained assuming the existence of a single defect on a single circuit node.

Moreover, it was shown that with $f = 10$, the coverage ranged from 81.37% to 99.54% for many industrial and ISCAS89 benchmark circuits. For $f = 20$, the coverage varied from 86.94% to 99.54%. For $f = 40$, the coverage ranged from 88.10% to 99.91%. With $f \leq 80$, the coverage varied from 94.82% to 100%. For more experimental results, refer to [38].

Finally, a maximum of 100000 faults where simulated for each fault multiplicity. It can be shown using probability theory [122] that such a number is sufficient for the simulation sizes above.

6.5.3 Simulation Results

The simulation experiments were conducted on a Sun SPARC-20 station with 256M of RAM, using the above simulation variables.

Tables 6.1- 6.7 contain a summary of these experiments and the comparison between the simulated results and those obtained analytically from Equations 6.24, 6.29, and 6.36.

Examining the above tables, the following can be concluded:

f	H	$\overline{N}_{\mathrm{DTE}}$	$N_{\mathrm{DTE}_{\max}}$	$N^{T}_{\mathrm{DTE}_{\max}}$	$N_{\mathrm{DTE}_{\min}}$	$N^{T}_{\mathrm{DTE}_{\min}}$	$\%\overline{DE}$
1	8.0000	8.0000	8	8	8	8	96.8750
2	14.9943	14.0297	15	15	8	8	94.5196
4	27.3810	24.3589	27	27	12	9	90.4848
8	48.5414	41.2508	47	47	26	12	83.8864
16	83.0595	67.7855	79	79	49	19	73.5212
32	135.4192	106.9987	122	127	84	34	58.2036
64	203.5669	159.7950	177	191	136	65	37.5800
128	251.6728	219.0796	235	255	198	128	14.4220

Table 6.2: Simulation results for $x = 256$, and $j = 100000$.

f	H	$\overline{N}_{\mathrm{DTE}}$	$N_{\mathrm{DTE}_{\max}}$	$N^{T}_{\mathrm{DTE}_{\max}}$	$N_{\mathrm{DTE}_{\min}}$	$N^{T}_{\mathrm{DTE}_{\min}}$	$\%\overline{DE}$
1	9.0000	9.0000	9	9	9	9	98.2421
2	16.9971	16.0213	17	17	9	9	96.8708
4	31.3980	28.2753	31	31	13	10	94.4774
8	56.6215	48.9855	55	55	31	13	90.4325
16	99.4082	82.8402	94	95	60	20	83.8202
32	168.9095	135.9931	155	159	107	35	73.4388
64	274.0736	214.4801	237	255	182	66	58.1093
128	410.7551	320.1423	346	383	287	129	37.4722

Table 6.3: Simulation results for $x = 512$, and $j = 100000$.

f	H	$\overline{N}_{\mathrm{DTE}}$	$N_{\mathrm{DTE}_{\max}}$	$N^{T}_{\mathrm{DTE}_{\max}}$	$N_{\mathrm{DTE}_{\min}}$	$N^{T}_{\mathrm{DTE}_{\min}}$	$\%\overline{DE}$
1	10.0000	10.0000	10	10	10	10	99.0234
2	18.9985	18.0040	19	19	10	10	98.2417
4	35.4065	32.2422	35	35	17	11	96.8513
8	64.6612	56.8223	63	63	37	14	94.4509
16	115.5799	98.3020	110	111	74	21	90.4002
32	201.6306	166.0285	186	191	136	36	83.7863
64	341.1039	272.3316	302	319	233	67	73.4051
128	551.8796	429.4989	464	511	384	130	58.0567
256	825.6300	640.8851	679	767	601	257	37.4136

Table 6.4: Simulation results for $x = 1024$, and $j = 100000$.

f	H	$\overline{N}_{\mathrm{DTE}}$	$N_{\mathrm{DTE}_{\max}}$	$N^{T}_{\mathrm{DTE}_{\max}}$	$N_{\mathrm{DTE}_{\min}}$	$N^{T}_{\mathrm{DTE}_{\min}}$	$\%\overline{DE}$
1	11.0000	11.0000	11	11	11	11	99.4629
2	20.9992	20.0037	21	21	11	11	99.0232
4	39.4108	36.2230	39	39	19	12	98.2313
8	72.6810	64.7391	71	71	41	15	96.8389
16	131.6651	113.9431	126	127	89	22	94.4364
32	233.9855	196.9139	218	223	161	37	90.3851
64	406.5697	332.4824	362	383	292	68	83.7655
128	685.9899	545.1437	588	639	497	131	73.3817
256	1107.9900	859.3920	911	1023	802	258	58.0375
512	1655.8791	1282.5091	1337	1535	1225	513	37.3775

Table 6.5: Simulation results for $x = 2048$, and $j = 100000$.

f	H	$\overline{N}_{\mathrm{DTE}}$	$N_{\mathrm{DTE}_{\max}}$	$N^{T}_{\mathrm{DTE}_{\max}}$	$N_{\mathrm{DTE}_{\min}}$	$N^{T}_{\mathrm{DTE}_{\min}}$	$\%\overline{DE}$
1	12.0000	12.0000	12	12	12	12	99.7070
2	22.9996	21.9920	23	23	12	12	99.4631
4	43.4129	40.2433	43	43	28	13	99.0175
8	80.6909	72.6731	79	79	53	16	98.2258
16	147.7075	129.7308	141	143	110	23	96.8327
32	266.1615	228.2099	247	255	193	38	94.4285
64	471.2910	394.0825	422	447	351	69	90.3788
128	816.9452	665.2962	706	767	619	132	83.7574
256	1376.2604	1090.5093	1142	1279	1033	259	73.3762
512	2220.7102	1719.6274	1782	2047	1650	514	58.0169
1024	3316.8768	2565.4072	2645	3071	2490	1025	37.3680

Table 6.6: Simulation results for $x = 4096$, and $j = 100000$.

f	H	$\overline{N}_{\mathrm{DTE}}$	$N_{\mathrm{DTE}_{\max}}$	$N^{T}_{\mathrm{DTE}_{\max}}$	$N_{\mathrm{DTE}_{\min}}$	$N^{T}_{\mathrm{DTE}_{\min}}$	$\%\overline{DE}$
1	13.0000	13.0000	13	13	13	13	99.8413
2	24.9998	23.9936	25	25	13	13	99.7071
4	47.4139	44.1704	47	47	31	14	99.4608
8	88.6958	80.6544	87	87	62	17	99.0155
16	163.7287	145.5745	157	159	121	24	98.2230
32	298.2492	259.7043	280	287	224	39	96.8298
64	535.6488	456.6789	488	511	413	70	94.4253
128	946.3993	788.3324	834	895	735	133	90.3768
256	1638.1947	1330.7712	1386	1535	1263	260	83.7552
512	2757.3007	2182.0434	2255	2559	2082	515	73.3637
1024	4446.6503	3439.2397	3522	4095	3345	1026	58.0171

Table 6.7: Simulation results for $x = 8192$, and $j = 100000$.

1. The average number of required diagnostic test experiments increases/decreases with the increase/decrease in the fault multiplicity, relative to the number of scan elements.

2. The average number of test experiments is upper-bounded by the entropy function H. Combining this fact with Equation 6.39, then,

$$\frac{1}{2}H < \overline{N}_{\mathrm{DTE}} \leq H \tag{6.46}$$

$$\frac{1}{2}\log_2\binom{x}{f} < \overline{N}_{\mathrm{DTE}} \leq \log_2\binom{x}{f} . \tag{6.47}$$

   Hence, the entropy function can be effectively used to obtain both an upper and a lower bound on the average number of required DTEs.

3. The simulation results confirm very strongly the validity of the developed analytical bounds for estimating the maximum and minimum number of diagnostic test experiments required for a complete diagnosis.

4. A high diagnostic efficiency is achieved over a wide range of fault multiplicities. The reduction in the number of BIST sessions, obtained with the adaptive algorithm, represents great savings in the diagnostic effort over the straightforward bottom line solution. Indeed, the simulation results indicate that for all values of f that can be considered practical, the scan elements being in error can be correctly identified in a number of test experiments which are only 1%-63% of x.

   For instance, for a 8192-bit scan channel containing 1024 discrepant scan cells, in order to diagnose any 1024-tuple of faulty scan cells with a 100% diagnostic resolution, the average number of DTEs is only 42% of the total number of cells in the channel.

5. It is clearly evident that the average diagnostic efficiency is a strong function of the ratio of discrepant scan channels/cells to the total number of scan elements.

In light of the last point, it is of interest to determine the relation between the number of scan elements and the average number of test experiments required to locate any multiplicity of faults.

To this end, the simulation results were analyzed in the $\log_2$-domain. Using simple numerical analysis, it was concluded that the expected diagnostic efficiency can be approximated by:

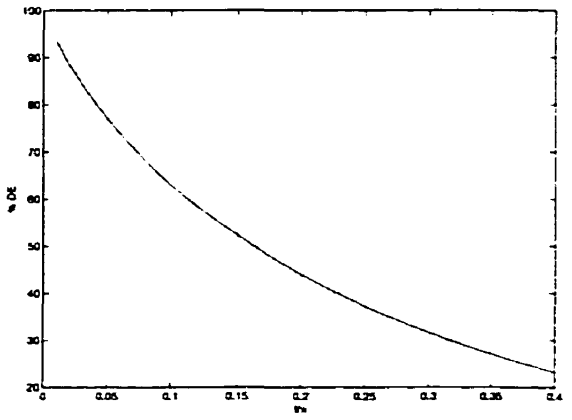


Figure 6.8: The expected diagnostic efficiency as a function of f/x.

$$\overline{DE} \approx 2^{-\alpha\beta^{\gamma}} \quad , \tag{6.48}$$

where $\alpha = 4.5104079$, $\beta = 0.5626896$, and $\gamma = -\log_2\left(\frac{f}{x}\right)$.

A plot of $\overline{DE}$ is shown in Figure 6.8. The figure strongly supports the viability of the adaptive algorithm as a practical BIST fault diagnosis technique.

Using Equation 6.40, the expected number of DTEs required to diagnose x scan elements containing f erroneous ones is equal to:

$$\begin{aligned} \overline{N}_{\mathrm{DTE}} &\approx x(1 - DE) && (6.49)\\ &= x\left(1 - 2^{-\alpha\beta^{\gamma}}\right) && (6.50)\\ &= x\left(1 - 2^{-\left[(4.5104079)(0.5626896)^{-\log_2\left(\frac{f}{x}\right)}\right]}\right) \quad . && (6.51) \end{aligned}$$

It should be emphasized that the above approximations hold for any fault multiplicity $f \leq \frac{x}{2}$. For example, for $x = 256$, the results are presented in Table 6.8 for different f, where $\overline{N}^{T}_{\mathrm{DTE}}$ and $\overline{DE}^{T}$ correspond to Equations 6.51 and 6.48, respectively.

Clearly, the simulation results yield very close values to the ones predicted by Equations 6.48 and 6.51 for $f \leq 0.4\,x$. These formulas, because of their simplicity, can be employed to quickly asses the required diagnostic effort, and the effectiveness of adaptive BIST fault diagnosis.

f	$\overline{N_{\mathrm{DTE}}}$	$\overline{N_{\mathrm{DTE}}}^{T}$	$\%\overline{DE}$	$\%\overline{DE}^{T}$
5	28.9374	28.8150	88.6963	88.7441
11	52.0365	52.5340	79.6732	79.4789
38	118.8951	121.3283	53.5566	52.6061
57	150.1600	151.8327	41.3437	40.6903
86	185.2071	183.7341	27.6535	28.2288
117	211.6998	205.9829	17.3050	19.5379

Table 6.8: Simulation vs. analytical results for $x = 256$, and $j = 100000$.

6.6 Concluding Remarks

In summary, the adaptive algorithm was presented in this chapter as an alternative BIST diagnostic technique.

A theoretical model of the algorithm was developed that bounds the number of diagnostic test experiments depending on the expected number of faulty scan channels and discrepant scan cells within a faulty channel. The yielded bounds, because of their simplicity, can be employed directly to asses the effectiveness of adaptive diagnosis and estimate the diagnostic effort. The analytical model was validated by a series of extensive simulation experiments.

The proposed algorithm employs a simple selection hardware which can be easily integrated in different BIST environments. It also uses signals from hardware components that can be shared with the normal BIST circuitry.

The algorithm improves the quality of BIST diagnosis. Subdividing the diagnostic process into a series of *guided* BIST sessions, with different compaction arrangements, reduces the probability of detective fault aliasing in consecutive BIST sessions, thus increasing the effective coverage of BIST.

The proposed algorithm also allows for flexible tradeoffs between the diagnosis time and the diagnostic hardware overhead.

Compared to the hardware overhead with recurring silicon cost for every manufactured module, the diagnosis time expenditure is just a one time cost applied only for the faulty MUTs that require diagnosis [38]. Such expenditure is significantly less than the labour-intensive development cost for the traditional fault diagnosis techniques.

The diagnosis time can be further reduced if more hardware overhead is allowed. Using multiple sets of the diagnostic hardware allows for a parallel diagnosis thus reducing the

diagnosis time. For example, using two sets of CSC and splitting the MISR during diagnosis, reduces the diagnosis time by half. The length of splited MISRs must be sufficient to yield a satisfactory detective fault aliasing probability.

The adaptive algorithm can also be extended to diagnose faults in a multiple frequency environment. For example in telecommunication applications, modules are often designed to operate at different frequencies in order to save power, reduce electro-magnetic interference, and reduce silicon area [63].

In such modules, each scan chain might operate from a separate clock domain. In practice, several chains are likely to share the same clock [63].

For the multiple frequency BIST described in [13, 63], the algorithm can be easily extended to this environment by using different copies of the diagnostic hardware, one for each frequency domain. If the hardware overhead is not acceptable, then a single copy of the diagnostic hardware can be used if the scan domains are diagnosed one at a time.

Finally, it must be mentioned that the adaptive algorithm works in any BIST environment that employs a *linear compactor* which preserves the empirical signature relations developed in Section 5.3. The algorithm may also provide significant savings with *non-linear* compactors as long as the total number of test experiments $N_{\mathrm{DTE}} + N_{\mathrm{DTE}^c}$ is smaller than the total number of scan channels or scan cells within a faulty channel.

Chapter 7

Non-Adaptive BIST Fault Diagnosis

In this chapter, another fault diagnosis algorithm is developed that can be used for the *non-adaptive* identification of the fault capturing scan channels/cells in a GSTUMPS-like module.

In this algorithm, the multiplicity of the faulty scan elements is fixed. An upper limit on this multiplicity is pre-assumed.

By incorporating the non-adaptive diagnostic scheme into the GSTUMPS architecture, common sources of defects can be identified in a series of diagnostic BIST sessions, combined with the post-diagnosis analysis described in Chapter 8.

The non-adaptive technique prescribes to the principle of *elimination*. It guarantees the identification of all erroneous scan elements. However, the algorithm *may* also produce some extra *false alarms*. If the actual number of faulty scan elements is greater than the assumed upper limit, the generation of false alarms is definite. It will be shown that the number of such alarms can be controlled at the expense of more diagnostic test time.

The non-adaptive algorithm has the advantage of minimizing the tester socket-time as compared to the adaptive algorithm which minimizes the number of executed diagnostic test experiments. It also allows for a high diagnostic resolution combined with the ability to tradeoff the quality of diagnosis for diagnosis time.

Like the adaptive algorithm, fault isolation is done by additional hardware, combined with some basic diagnostic software support. The non-adaptive technique also employs the concepts of signature superposition, thus requiring a minimal number of reference signatures to be stored. Hence, the computational space and time for fault isolation are

reduced substantially.

The proposed scheme does not assume any specific fault model and thus, it can diagnose all voltage-detectable faults, including delay faults [3], irrespective of their multiplicities.

Several measures have been defined to quantify the diagnostic effectiveness of the algorithm. Simulation experiments with various modules, having different number of scan channels/cells and assuming different number of error-capturing scan elements, have been used to quantify the performance of the non-adaptive algorithm.

In Section 7.1, the general concepts of the non-adaptive algorithm are described. An analytical model of the algorithm is developed in Section 7.2. The selection of scan elements to be observed in each diagnostic test experiment is explained in Section 7.3. Section 7.4 describes the details of identifying the fault-capturing scan channels and the required diagnostic hardware. Section 7.5 extends the algorithm to diagnose the faulty scan cells, and provides the needed diagnostic hardware. The performance of the non-adaptive algorithm is analyzed in Section 7.6. Finally, Section 7.7 reports the experimental results and validates the predicted analytical performance.

7.1 Principles of the Non-Adaptive Algorithm

In this section, the general principles and concepts of non-adaptive diagnosis are presented.

Like the adaptive approach, the non-adaptive algorithm has two modes of operation. In MSC mode, the algorithm is employed to identify the faulty scan channels. Each of the identified faulty channels becomes the subject of diagnosis in SSC mode, where the discrepant scan cells within each of the identified channels are located. The algorithm is assumed to be implemented in system diagnostic software.

The complete process of diagnosis consists of a number of test experiments. The algorithm is non-adaptive in the sense that all test experiments are *predetermined* before starting the diagnosis process, assuming a certain fault multiplicity (f).

The DTEs are executed in a sequential manner, and their results are analyzed at the end of the diagnosis process to determine all faulty scan elements of multiplicity f or less. The value of f can be determined by analyzing the MUT structure under certain fault model(s).

The off-line *reasoning* analysis used with non-adaptive diagnosis is based on the principle of elimination (divide and conquer).

Each DTE identifies the *possibly* faulty or fault-free scan elements by examining the compacted signature. If the signature produced by a set of scan elements is error-free, neglecting the controllable impact of traditional signature aliasing, then all of these elements can be declared fault-free and can be eliminated from further analysis. This restricts the search for the discrepant elements to the remaining scan elements.

On the other hand, if the compacted signature is erroneous, then at least one of the observed elements is corrupted by a fault. This continues until all DTE results are analyzed. Once the post-test reasoning analysis is complete, the set of scan elements whose status are not determined to be error-free are considered to be erroneous scan element *candidates*.

The diagnostic information needed for the off-line reasoning analysis is maintained in the form of a list of fault-free signatures, one for each DTE. Like the adaptive algorithm, only the reference signatures of the scan cells are actually generated. All other signatures are obtained from the latter ones using the principle of signature superposition and the analytical results developed in Section 5.3.

Due to its non-adaptive nature, the algorithm is subject to *diagnostic aliasing*. Diagnostic aliasing can manifest itself in two forms.

First, it is possible for the algorithm to falsely declare error-capturing scan elements as being fault-free. This is similar to the *type-1 aliasing* defined in [111]. Since the set of all diagnostic test experiments is guaranteed to involve all scan elements, the identification of the faulty ones is guaranteed. Hence, the probability of such aliasing is zero, neglecting the effects of traditional signature aliasing.

Second, it is possible for the algorithm to falsely declare error-free scan elements as error capturing ones. In this case, the list of candidate faulty scan elements is a *superset* of the list of actual failing ones. Thus, the candidate list includes good scan elements as well as failing ones. This can happen if the actual number of failing elements is greater than the assumed f, or the executed DTEs are not sufficient for a complete diagnosis. This type of aliasing is similar to the *type-2 aliasing* defined in [111].

A type-2 aliasing is not severe in the sense that a post-diagnosis analysis, with classical diagnostic methods, will reveal that no errors were captured at the good scan elements identified as being faulty. Thus, they can be classified as *false alarms*. However, the number of produced false alarms will directly impact the amount of time spent in the

post-diagnosis analysis, in order to isolate the exact cause of a failure.

Hence, the scan elements in each diagnostic test experiment must be selected such that the maximum number of fault-free cells can be eliminated. This in turn will reduce the number of false alarms. The possibility of such alarms is explored statistically in Section 7.6, and is then verified with simulation experiments in Section 7.7.

7.2 Analytical Model of Non-Adaptive Diagnosis

In this section, an analytical model of the non-adaptive diagnosis process is developed. It relates the number of test experiments to the number of faulty scan channels or discrepant scan cells within a faulty channel.

The proposed model decides the subset of scan elements that must be made observable to the MISR in each DTE. In this model, the complete process of diagnosis consists of executing a number of Diagnostic Test Experiment Groups (DTEGs). Each DTEG consists of "b" test experiments that capture test responses from *disjoint* subsets of scan elements. Thus, each scan channel/cell is associated with a single DTE for any given DTEG.

Clearly, the finest diagnostic resolution can be obtained with a reasonable diagnosis time, provided that the partitions provided by the DTEs in different DTEGs are orthogonal in nature.

Without loss of generality, it is assumed that the number of scan elements associated with each DTE is approximately x/b, where x is the total number of scan elements.

The non-adaptive diagnostic procedure has the following properties:

- During each DTEG, every scan element is assigned an integer *label* b_i, where $0 \leq b_i < b$.
- During DTE_i, only the scan elements with a label matching b_i, are made observable to the MISR.
- Once the whole set of "b" DTEs are executed, a new DTEG is generated corresponding to a new partition of the scan elements, until all DTEGs are applied.

Theorem 7.2.1 *The average number of scan elements ($\bar{x}_{EF_G}$) that can be declared as error-free, after executing a single diagnostic test experiment group, is given by [162]:*

$$\bar{x}_{EF_G} = x\left(1 - \frac{1}{b}\right)^f \quad , \tag{7.1}$$

where f is the maximum number of scan elements affected by a fault, b is the number of test experiments in the group G, and x is the number of scan elements.

Proof: As assumed previously, the average number of scan elements involved in any DTE is approximately x/b. If none of them capture errors during test, then all x/b scan elements can be declared as error-free. On the other hand, if the DTE involves at-least a single discrepant scan element, then none of the x/b elements can be declared as error-free.

Let p be the probability that a DTE does not involve any error-capturing scan elements. Then, $1-p$ is the probability that a DTE involves at-least a single discrepant scan element.

Let f be the maximum number of scan elements that can be affected by a fault during any DTE. Assuming a *uniform* probability distribution, then p can be expressed as the ratio of all possible *distinct* $(\frac{x}{b})$-element combinations, that do not contain any of the fault-capturing scan elements, to the number of all possible *distinct* $(\frac{x}{b})$-element combinations chosen from all x scan elements. Thus,

$$p = \frac{\binom{x-f}{x/b}}{\binom{x}{x/b}} \tag{7.2}$$

$$= \frac{(x-f)!/(x-f-x/b)!(x/b)!}{x!/(x-x/b)!(x/b)!} \tag{7.3}$$

$$= \frac{(x-f)!}{(x-f-x/b)!}\left[\frac{(x-x/b)!}{x!}\right] \tag{7.4}$$

$$= \frac{(x-f)!}{(x-f-x/b)!}\left[\frac{(x-f-x/b)!\prod_{i=0}^{f-1}(x-i-x/b)}{(x-f)!\prod_{i=0}^{f-1}(x-i)}\right] \tag{7.5}$$

$$= \frac{\prod_{i=0}^{f-1}(x-i-x/b)}{\prod_{i=0}^{f-1}(x-i)} \tag{7.6}$$

$$= \prod_{i=0}^{f-1}\left[\frac{x-i-x/b}{x-i}\right] \tag{7.7}$$

$$= \prod_{i=0}^{f-1}\left[\frac{x-i}{x-i} - \frac{x/b}{x-i}\right] \tag{7.8}$$

$$= \prod_{i=0}^{f-1}\left[1 - \frac{x}{b(x-i)}\right] . \tag{7.9}$$

Since $x >> f$ in practical real-life modules, then $x/(x-i) \approx 1$, where $0 \leq i < f$. Thus,

$$p \approx \prod_{i=0}^{f-1}\left[1 - \frac{1}{b}\right] \tag{7.10}$$

$$= \left(1 - \frac{1}{b}\right)^{f} . \tag{7.11}$$

The expected number of scan elements ($\bar{x}_{EF}$) that can be declared as error-free after executing a single DTE is equal to:

$$\bar{x}_{EF} = \frac{x}{b} \times p + 0 \times (1-p) \tag{7.12}$$

$$= \frac{x}{b}\left(1 - \frac{1}{b}\right)^{f} . \tag{7.13}$$

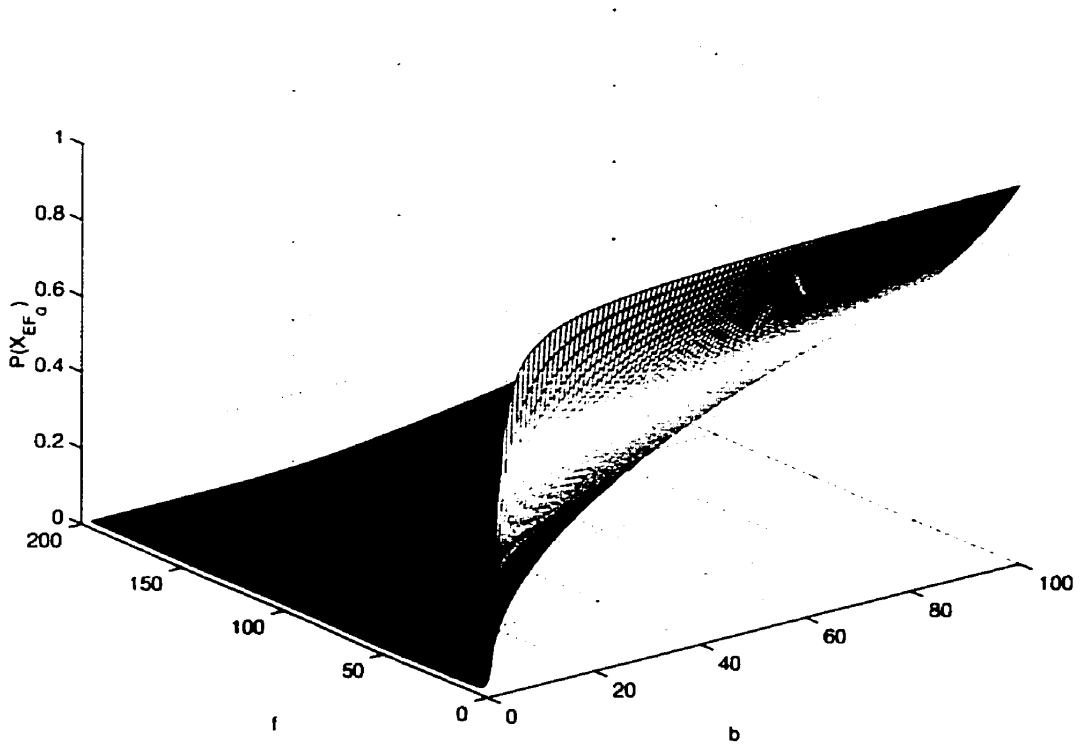


Figure 7.1: The probability $P(x_{EF_G})$ as a function of f and b.

Since the b DTEs in a test group are mutually exclusive in terms of the scan elements they involve, then the average number of scan elements that can be declared as error-free after executing a single group is equal to:

$$\bar{x}_{EF_G} = b \times \bar{x}_{EF} \tag{7.14}$$

$$= x\left(1 - \frac{1}{b}\right)^f \quad . \tag{7.15}$$

Hence the theorem is proved □.

It is interesting to note that Equation 7.15 has the form $\bar{x}_{EF_G} = x \cdot P(x_{EF_G})$, where $P(x_{EF_G}) = (1 - 1/b)^f$ is the probability that a randomly chosen scan element will be declared error-free after executing a single DTEG. A plot of $P(x_{EF_G})$ is shown in Figure 7.1.

Clearly, $P(x_{EF_G})$ increases as the number of executed DTEs per group is increased, and decreases as the number of discrepant scan elements increases.

As stated in Section 7.1, the scan elements in each test experiment must be selected

such that the maximum number of fault-free elements can be eliminated. A diagnostic test experiment that achieves the maximal fault resolution is said to be an *optimal* DTE.

Theorem 7.2.2 *The average number of scan elements ($\bar{x}_{EF}$) that can be declared as error-free, after executing a single optimal diagnostic test experiment, is maximized for $b = f+1$, where f is the maximum number of scan elements affected by a fault, b is the number of diagnostic test experiments in the group, and x is the number of scan elements [162].*

Proof:

In order to maximize the average number of scan elements that can be declared as error-free, after executing a single DTE, $\bar{x}_{EF}$ must be differentiated with respect to b and equated to zero. Let,

$$
\begin{aligned}
0 &= \frac{\partial \bar{x}_{EF}}{\partial b} && (7.16)\\
&= \frac{\partial}{\partial b}\left[\frac{x}{b}\left(1-\frac{1}{b}\right)^{f}\right] && (7.17)\\
&= -\frac{x}{b^2}\left(1-\frac{1}{b}\right)^{f} + \frac{x}{b^2}\left(1-\frac{1}{b}\right)^{f-1}\left(\frac{f}{b}\right) && (7.18)\\
&= \frac{x}{b^2}\left(1-\frac{1}{b}\right)^{f-1}\left[\frac{f}{b} - \left(1-\frac{1}{b}\right)\right] && (7.19)\\
&= \frac{x}{b^2}\left(1-\frac{1}{b}\right)^{f-1}\left[\frac{f+1}{b} - 1\right] . && (7.20)
\end{aligned}
$$

Since $x, f > 0$ and $b > 1$, then,

$$\frac{f+1}{b} - 1 = 0 \ , \qquad (7.21)$$

and $b = f + 1$.

In order to check that $b = f + 1$ results indeed in a maximum, the second derivative of $\bar{x}_{EF}$ must be taken with respect to b:

$$\frac{\partial^2 \bar{x}_{EF}}{\partial b^2} = \frac{\partial}{\partial b}\left[\frac{x}{b^2}\left(1-\frac{1}{b}\right)^{f-1}\left(\frac{f+1}{b}-1\right)\right] \tag{7.22}$$

$$= \left[-\frac{2x}{b^3}\left(1-\frac{1}{b}\right)^{f-1}+\frac{x(f-1)}{b^4}\left(1-\frac{1}{b}\right)^{f-2}\right]\left(\frac{f+1}{b}-1\right)$$
$$-\frac{x(f+1)}{b^4}\left(1-\frac{1}{b}\right)^{f-1} . \tag{7.23}$$

Substituting $b = f + 1$ into the second derivative and noting that $x, f > 0$,

$$\left.\frac{\partial^2 \bar{x}_{EF}}{\partial b^2}\right|_{b=f+1} = -\frac{x(f+1)}{(f+1)^4}\left(1-\frac{1}{f+1}\right)^{f-1} \tag{7.24}$$

$$= -\frac{x}{(f+1)^3}\left(\frac{f}{f+1}\right)^{f-1} \tag{7.25}$$

$$< 0 . \tag{7.26}$$

Hence, $b = f + 1$ indeed results in a maximum, and the theorem is proved □.

The last theorem constitutes the basis on which the selection of scan elements in each DTE is done. The generation of such DTEs is considered next.

7.3 Non-Adaptive DTE Generation

In this section, a systematic method is presented for selecting the scan elements involved in each diagnostic test experiment.

For the sake of simplicity, assume for the moment that the total number of scan elements x is an integer power of $b = f + 1$, i.e., $x = b^g$ for some integer $g \geq 1$. Thus, the number of scan elements associated with each DTE is exactly equal to $\frac{x}{b}$.

During each DTEG, every scan element is assigned an integer *label* b_i, $0 \leq b_i < b$. DTE_i involves all scan elements with a label matching b_i.

The problem that must be solved is that of assigning labels to the scan elements in each group, such that the DTEs in different groups are *orthogonal* in nature. This allows an optimum diagnostic resolution to be achieved with a minimal number of test groups.

The label assignment problem is analogous to that of devising a code. Accordingly, some basic terminology and definitions are given first. The terminology and notations used are similar to those used in Coding Theory [84].

Definition 7.3.1 *A set is an arbitrary collection of elements, without any predefined operations between the set elements. A set is characterized by its cardinality, which is defined as the number of elements contained in it.*

Corollary 7.3.1 *The integers* $\{0, 1, \ldots, b-1\}$ *form a set* $\mathcal{B}$ *with cardinality* $|\mathcal{B}| = b$.

Definition 7.3.2 *A commutative group is a set of elements* $\mathcal{G}$ *on which the binary operation* $\otimes$ *is defined. The operation must satisfy the following requirements:*

1. *Closure:* $(u_i \otimes u_j) = u_k \in \mathcal{G}$ *for all* $u_i, u_j \in \mathcal{G}$.
2. *Associativity:* $(u_i \otimes u_j) \otimes u_k = u_i \otimes (u_j \otimes u_k)$ *for all* $u_i, u_j, u_k \in \mathcal{G}$.
3. *Commutativity: for all* $u_i, u_j \in \mathcal{G}$, $u_i \otimes u_j = u_j \otimes u_i$.
4. *Identity: there exists* $I \in \mathcal{G}$ *such that* $u_i \otimes I = I \otimes u_i = u_i$ *for all* $u_i \in \mathcal{G}$.
5. *Inverse: for all* $u_i \in \mathcal{G}$ *there exists a unique element* $u_i^{-1} \in \mathcal{G}$ *such that* $u_i \otimes u_i^{-1} = u_i^{-1} \otimes u_i = I$.

Theorem 7.3.1 *The set* $\mathcal{B} = \{0, 1, \ldots, b-1\}$ *forms a commutative group of order* b *under modulo-*b *integer addition for any positive integer* b.

Proof: First, closure is assured by the modularity of the additive operation. Since integer addition is associative and commutative, then mod $-\ b$ addition is also associative and commutative. The identity element in $\mathcal{B}$ is clearly 0. Finally, the mod $-\ b$ inverse of an element b_i is the integer $b - b_i$, and the theorem is proved □.

Definition 7.3.3 *Two integers* u_i *and* u_j *are said to be in the same equivalence class mod-*M *addition, if* u_i *can be written as* $u_i = u_k M + u_j$ *for some integer* u_k.

Elements in an equivalence class mod $-\ M$ are *equivalent* in the sense that any element in a given class can be substituted for any other element in the same class without changing the outcome of a mod $-\ M$ operation. Equivalence classes of integers are usually *labeled* with their *smallest* constituent nonnegative integer.

Corollary 7.3.2 *Addition modulo-b groups the set of scan elements* $\mathcal{X} = \{0, 1, \ldots, x-1\}$ *into b distinct classes with cardinality* $|\frac{x}{b}|$, *where* $x = b^g$. *Each class has an integer label* b_i *associated with it, where* $0 \leq b_i < b$.

Theorem 7.3.2 *The set* $\mathcal{B} - \{0\} = \{1, 2, \ldots, b-1\}$ *forms a commutative group of order* $(b-1)$ *under modulo-b multiplication if and only if b is a prime integer.*

Proof: Multiplication mod $-$ b is performed over integers in the same manner as mod $-$ b addition. The result of a regular integer multiplication operation is divided by the modulus b, and the positive remainder is retained as the result of the modular multiplication.

The modular multiplication operation derives its commutativity and associativity from those of regular integer multiplication. The multiplicative identity is clearly 1. However, closure and the existence of an inverse for all elements, is only guaranteed if b is prime.

If b is not prime, then there exists distinct $b_i, b_j \in \mathcal{B} - \{0\}$ such that $1 \leq b_i, b_j < b$ and $b_i b_j \equiv 0 \bmod b$. Thus, closure is not satisfied. On the other hand, if b is prime, then there cannot be such pair of elements and closure is satisfied.

Furthermore, given an element $b_i \in \mathcal{B} - \{0\}$, the products $\{b_i \cdot 1, b_i \cdot 2, \ldots, b_i \cdot (b-1)\}$ must be distinct. Otherwise, $b_i b_j = b_i b_k$, implying $b_i(b_j - b_k) \equiv 0 \notin \mathcal{B} - \{0\}$. Since the $(b-1)$ products are distinct, one of them must be equal to the multiplicative identity "1", thus assuring the existence of inverses for all $b_i \in \mathcal{B} - \{0\}$ $\square$.

Corollary 7.3.3 *Multiplication modulo-b groups the set of scan elements* $\mathcal{X} = \{0, 1, \ldots, x-1\}$ *into b distinct classes with cardinality* $|\frac{x}{b}|$, *where* $x = b^g$. *Each class has an integer label* b_i *associated with it, where* $0 \leq b_i < b$.

The last corollary follows from the fact that the $(b-1)$ products $\{x_i \cdot 1, x_i \cdot 2, \ldots, x_i \cdot (b-1)\}$ are distinct. Combining Corollary 7.3.2 and Corollary 7.3.3, it is clear that in any diagnostic test experiment group, mod $-$ b addition and multiplication can be used to obtain the partitions of scan elements associated with each of its b test experiments. What remains to be determined is a systematic way of obtaining distinct DTEGs that are orthogonal in nature. To this end, additional mathematical background needs to be established first.

Definition 7.3.4 *Let $\mathcal{F}$ be a finite set of elements on which the two operations $\otimes$ and $\oslash$ are defined. $\mathcal{F}$ is said to be a Galois Field of order $|\mathcal{F}|$ (GF($|\mathcal{F}|$)), if and only if:*

1. *$\mathcal{F}$ forms a commutative group under $\otimes$. The $\otimes$ identity element is labeled "0".*
2. *$\mathcal{F} - \{0\}$ forms a commutative group under $\oslash$. The $\oslash$ identity element is labeled "1".*
3. *The operations $\otimes$ and $\oslash$ distribute: $u_i \oslash (u_j \otimes u_k) = (u_i \oslash u_j) \otimes (u_i \oslash u_k)$.*

Theorem 7.3.3 *The set $\mathcal{B} = \{0, 1, \ldots, b-1\}$, where b is a prime, forms the field* GF(b) *under modulo-b addition and multiplication.*

Proof: By Theorem 7.3.1, the set $\mathcal{B} = \{0, 1, \ldots, b-1\}$ forms a commutative group under mod $-\ b$ addition. By Theorem 7.3.2, the set $\mathcal{B} - \{0\}$ forms a commutative group under mod $-b$ multiplication. Finally, the two operations distribute in integer arithmetic, and the theorem is proved □.

Definition 7.3.5 *Let $\mathcal{V}$ be the set of elements called vectors, and $\mathcal{F}$ a field of elements called scalars. Let "+" denote the vector addition operation, that maps pairs of vectors $\mathbf{v}_i, \mathbf{v}_j \in \mathcal{V}$ into a vector $\mathbf{v}_k = \mathbf{v}_i + \mathbf{v}_j \in \mathcal{V}$. Let "$\cdot$" denote the scalar multiplication operation, that maps a scalar $u_i \in \mathcal{F}$ and a vector $\mathbf{v}_j \in \mathcal{V}$ into another vector $\mathbf{v}_k = u_i \cdot \mathbf{v}_j \in \mathcal{V}$. $\mathcal{V}$ is said to form a vector space over $\mathcal{F}$, if the following conditions are satisfied:*

1. *$\mathcal{V}$ forms a commutative group under the operation "+".*
2. *For any element $u_i \in \mathcal{F}$ and $\mathbf{v}_j \in \mathcal{V}$, $u_i \cdot \mathbf{v}_j = \mathbf{v}_k \in \mathcal{V}$.*
3. *The operations "+" and "$\cdot$" distribute: $u_i \cdot (\mathbf{v}_j + \mathbf{v}_k) = u_i \cdot \mathbf{v}_j + u_i \cdot \mathbf{v}_k$, and $(u_i + u_j) \cdot \mathbf{v}_k = u_i \cdot \mathbf{v}_k + u_j \cdot \mathbf{v}_k$.*
4. *For all $u_i, u_j \in \mathcal{F}$ and all $\mathbf{v}_k \in \mathcal{V}$, $(u_i \cdot u_j) \cdot \mathbf{v}_k = u_i \cdot (u_j \cdot \mathbf{v}_k)$.*
5. *The multiplicative identity "1" in $\mathcal{F}$ acts as a multiplicative identity in scalar multiplication. Thus, for all $\mathbf{v}_i \in \mathcal{V}$, $1 \cdot \mathbf{v}_i = \mathbf{v}_i$.*

Theorem 7.3.4 *Let $\mathcal{G}$ be the set of all x-tuple vectors $\mathbf{g}_i = < g_{i,0}, g_{i,1}, \ldots, g_{i,x-1} >$, where $0 \leq g_{i,j} < b$. $\mathcal{G}$ forms a vector space over $\mathcal{B} = \{0, 1, \ldots, b-1\}$ under mod-b addition and multiplication, for all prime b.*

The proof of the above theorem is trivial and follows directly from Theorem 7.3.3 and Definition 7.3.5.

Definition 7.3.6 *Two h-tuple vectors, $\mathbf{v}_i, \mathbf{v}_j \in \mathcal{V}$ are said to be orthogonal if their inner product, $\mathbf{v}_i \cdot \mathbf{v}_j = \sum_{k=0}^{h-1} v_{i,k} \cdot v_{j,k}$, is equal to zero.*

Theorem 7.3.5 *The set of vectors $\{\mathbf{g}_i\}$ in the space $\mathcal{G}$, each having $\frac{x}{b}$ elements associated with every $b_k \in \mathcal{B}$, forms an orthogonal subspace.*

Proof: The proof is trivial and follows directly from the properties of modular addition and multiplication. It must be shown that the inner product of any pair of vectors $\mathbf{g}_i, \mathbf{g}_j \in \mathcal{G}$ equals zero. The inner product is equal to:

$$\mathbf{g}_i \cdot \mathbf{g}_j = \left[\sum_{l=0}^{x-1} (g_{i,l} \cdot g_{j,l}) \bmod b\right] \bmod b \ . \tag{7.27}$$

However, since $\mathbf{g}_i, \mathbf{g}_j$ are assumed to have exactly $\frac{x}{b}$ elements associated with each $b_k \in \mathcal{B}$, then the set $\{(g_{i,l} \cdot g_{j,l}) \bmod b\}$ will also have exactly $\frac{x}{b}$ elements associated with each $b_k \in \mathcal{B}$. Since x was assumed at the beginning of this section to be an integer power of b, $x = b^g$ for some integer $g \geq 1$, then,

$$\begin{aligned}
\mathbf{g}_i \cdot \mathbf{g}_j &= \left[\sum_{k=0}^{b-1} \left(\frac{x}{b} \cdot b_k\right) \bmod b\right] \bmod b && (7.28)\\
&= \left[\sum_{k=0}^{b-1} \left(b^{g-1} \cdot b_k\right) \bmod b\right] \bmod b && (7.29)\\
&= \left[\sum_{k=0}^{b-1} 0\right] \bmod b && (7.30)\\
&= 0 \quad \square \ . && (7.31)
\end{aligned}$$

Recapping all the results obtained thus far, the partitions of scan elements associated with each of the b diagnostic experiments in any test group can be obtained using mod $-\,b$ addition and multiplication over the entire set of scan elements (x). If the number of scan elements is an integer power of b, then different test groups have an orthogonal number of

scan elements in common. What remains to be determined is a systematic way of obtaining these groups.

The operations defined in vector spaces allow one to compute *linear combinations* of vectors. Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_i$ be vectors in $\mathcal{V}$, and let $u_1, u_2, \ldots, u_i$ be scalars in $\mathcal{F}$. Since $\mathcal{V}$ forms a commutative group under "+", the linear combination $\mathbf{v} = u_1 \cdot \mathbf{v}_1 + u_2 \cdot \mathbf{v}_2 + \cdots + u_i \cdot \mathbf{v}_i$ is also a vector in $\mathcal{V}$.

Definition 7.3.7 *The collection of vectors* $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_i\}$, *the linear combinations of which include all vectors in a vector space* $\mathcal{V}$, *is said to be a spanning set for* $\mathcal{V}$.

Definition 7.3.8 *A spanning set for* $\mathcal{V}$ *having a minimal cardinality is called a basis for* $\mathcal{V}$. *If a basis has k vectors, then the vector space is said to be of dimension k.*

It follows immediately from the definition that the vectors forming a basis must be linearly independent. Otherwise, one of the vectors can be deleted, reducing the basis cardinality by one. Although a vector space may have several possible basis, all of them are equivalent, have the same cardinality, and can be obtained from each other using transformations.

Theorem 7.3.6 *The set of vectors* $\mathbf{g}_i =< g_{i,0}, g_{i,1}, \ldots, g_{i,x-1} >$, $0 \leq i < \log_b x$, *forms a basis for* $\mathcal{G}$ *of dimension* $\log_b x$, *where* $0 \leq g_{i,j} < b$,

$$g_{0,j} = j \bmod b \ , \tag{7.32}$$

$$q_{0,j} = j/b \ , \tag{7.33}$$

$$g_{i,j} = q_{i-1} \bmod b \ , \tag{7.34}$$

and,

$$q_{i,j} = q_{i-1}/b \ . \tag{7.35}$$

Proof: Consider the number representation of each element in the set $\mathcal{X} = \{0, 1, \ldots, x-1\}$ in base-b. Using Number Theory, a *minimum* of $\log_b x$ digits are required to represent each element in $\mathcal{X}$. Let the i^{th} digit for the j^{th} element be equal to $g_{i,j}$. Then, every element in $\mathcal{X}$ can be represented using the above recursive equations, and the theorem is proved □.

The above theorem clearly shows how the first $\log_b x$ distinct test experiment groups can be obtained. These test groups will be referred to as the basis-DTEGs. The remaining ones can be obtained with the following theorem.

Theorem 7.3.7 *Let* $\{\mathbf{g}_i\}_{i=0}^{\log_b x-1}$ *be a basis for the vector space* $\mathcal{G}$. *Then for each vector* $\mathbf{g} \in \mathcal{G}$ *there exists a unique combination (c) of* $\log_b x$*-scalars in* $\mathcal{B}$ *such that:*

$$\mathbf{g} = \sum_{i=0}^{\log_b x-1} b_{c,i}\mathbf{g}_i \ . \tag{7.36}$$

Proof: Since the basis spans $\mathcal{G}$, the existence of such combination of scalars is guaranteed by Definition 7.3.7. To prove the uniqueness, let o be another combination with:

$$\mathbf{g} = \sum_{i=0}^{\log_b x-1} b_{o,i}\mathbf{g}_i \ . \tag{7.37}$$

Thus,

$$\mathbf{0} = \mathbf{g} - \mathbf{g} \tag{7.38}$$

$$= \sum_{i=0}^{\log_b x-1} (b_{c,i} - b_{o,i})\mathbf{g}_i \ . \tag{7.39}$$

Since basis vectors are nonzero, then $b_{c,i} = b_{o,i}$ for all i, and the theorem is proved □.

Hence, the remaining diagnostic test experiment groups can be obtained as linear combinations of the basis-DTEGs. Clearly, the number of such combinations is equal to $|\mathcal{B}|^{\log_b x} = x$. The question remains as to whether all the linear combinations are distinct mod $-\ b$. This question is answered by the following theorem.

Theorem 7.3.8 *The total number of possible distinct diagnostic test experiment groups is equal to:*

$$N_{\text{DTEG}} = \frac{b^{\log_b x} - 1}{b - 1} \ , \tag{7.40}$$

where x *is the number of scan elements, and* b *is the number of test experiments in a group.*

Proof: By Theorem 7.3.6, there are $\log_b x$ *basis*-DTEGs. Consider the first test group ($\mathbf{g}_0$) in the basis. There is only one distinct way of partitioning the scan elements in $\mathbf{g}_0$

under mod $-b$ operations. Addition and multiplication mod $-b$ merely changes the labels associated with each DTE in the group. Two test groups differing only by a permutation of the labels associated with each DTE are characterized by the same partitions, and are considered to be *equivalent* test experiment groups.

For example, let $x = 9$ and $b = 3$. There are $\log_b x = \log_3 9 = 2$ basis DTEGs. However, $\mathbf{g}_0$, $(2 \cdot \mathbf{g}_0) \bmod 3$, $(\mathbf{g}_0 + 1) \bmod 3$, and $(\mathbf{g}_0 + 2) \bmod 3$ are all equivalent under mod -3 operations.

Next, consider the second test group ($\mathbf{g}_1$) in the basis. A simple counting exercise shows that there are additional b distinct ways of partitioning the scan elements using both $\mathbf{g}_1$ and $\mathbf{g}_0$. Thus, $1 + b$ distinct DTEGs can be obtained using both $\mathbf{g}_0$ and $\mathbf{g}_1$.

Using the above example, the additional DTEGs can be obtained as $\mathbf{g}_1$, $(\mathbf{g}_1 + \mathbf{g}_0) \bmod 3$, $((2 \cdot \mathbf{g}_1) \bmod 3 + \mathbf{g}_0) \bmod 3$. All other possible combinations are equivalent to the preceding ones.

Hence, there are additional b^i distinct ways of partitioning the scan elements using $\mathbf{g}_0, \ldots, \mathbf{g}_{i-1}$, $\mathbf{g}_i$, and,

$$N_{\text{DTEG}} = \sum_{i=0}^{\log_b x - 1} b^i \tag{7.41}$$

$$= \frac{1 - b^{\log_b x}}{1 - b} \quad \square \,. \tag{7.42}$$

The DTEG generation process is now complete. The procedure can be summarized as follows:

1. Obtain the basis-DTEGs using Theorem 7.3.6.
2. Generate the remaining DTEGs as distinct linear combinations of the basis-DTEGs using mod $-b$ addition and multiplication.

To conclude, some final remarks. In developing the DTEG generation procedure, it was assumed that the number of DTEs in a group is *prime*. One cannot use modular arithmetic in the construction of GF(b) if b is not a prime, for as noted earlier, the integers $\{1, 2, \ldots, b-1\}$ do not form a group under mod $-b$ multiplication.

Finite fields of prime order are quite easy to construct. But this leaves a large range of possible field orders for which a construction has not been described. It can be shown

that finite fields of order b^g, b a positive prime and $g \geq 1$, can be constructed as vector spaces over the prime order field GF(b), using methods that are somewhat more complex than simple modular arithmetic [157, 163, 164].

The procedure for generating DTEGs can still be used with b not a prime integer. The implication being that of a reduced number of possible distinct test groups. This follows from the fact that under mod $-$ b multiplication, the set $\{1, 2, \ldots, b-1\}$ will contain several *zero divisors*.

A zero divisor is any nonzero number b_i for which there exits another nonzero b_j such that $b_i \cdot b_j \equiv 0 \mod b$. In general, if the modulus b has factors other than 1 in a given set, the set will have zero divisors under mod $-$ b multiplication. For example, the set $\{0, 1, 2, 3, 4, 5, 6, 7\}$ under mod $-$ 8 multiplication, contains two zero divisors since $0 \equiv 0 \mod 8$ and $0 \equiv 2 \cdot 4 \mod 8$.

Finally, the number of scan elements has been assumed so far to be an integer power of b. The DTEG generation procedure can still be applied if the latter is not satisfied. The implication being that the number of scan elements involved in each DTE will not be equal, thus resulting in non-orthogonal test groups. This also requires a more complex hardware to implement the non-adaptive algorithm. The derived equations can still be used, with $\log_b x$ being replaced with $\lceil \log_b x \rceil$.

7.4 Identifying the Faulty Scan Channels

In this section, the working procedure of the non-adaptive algorithm in MSC-mode is explained. The procedure can be used in locating the faulty scan channels in a GSTUMPS module. It will be shown that hardware required to implement this scheme comprises simple components.

As mentioned in Section 7.2, the complete process of diagnosis consists of executing a number of pre-determined diagnostic test experiment groups (g_c), each comprising $b_c = f_c + 1$ test experiments, where f_c is the assumed maximum faulty multiplicity of the scan channels.

During each DTE, a subset of channels is made observable to the MISR. The complete set of T pseudorandom test patterns are applied to the MUT through the channels, and the test responses are shifted out to the compactor. The compacted signature is unloaded

into the tester, and a new DTE is executed.

The DTEs are executed in a sequential manner, and their results are analyzed off-line at the end of the diagnosis process, to determine all the *possibly* faulty scan channels.

Up to this point, no use was made of the fact that the test experiments in any group monitor disjoint subsets of the channels, and that each scan channel is monitored in a single DTE. Thus, once the signatures $\{S_{i,0}, S_{i,1}, \ldots, S_{i,b_c-2}\}$ for the first $b_c - 1$ DTEs in group i are known, the signature S_{i,b_c-1} for the b_c^{th} DTE can be obtained using Theorem 5.3.2 and Equation 5.25:

$$S_{i,b_c-1} = S + \sum_{j=0}^{b_c-2} S_{i,j} \ , \tag{7.43}$$

where S is the compacted signature from all channels at the beginning of diagnosis.

Hence, only $b_c - 1$ diagnostic experiments need to be evaluated explicitly in any test group. The remaining *complementary* one can be obtained using Equation 7.43. This translates into savings in the diagnostic effort equivalent to g_c test experiments.

Without loss of generality, it is assumed that the number of scan cells in each channel is n, where n is the number of scan cells in the longest channel. Furthermore, for the exactness of results, the number of scan channels (m) is assumed to to be an integer power of b_c. Thus, the number of scan channels associated with each DTE is equal to $\frac{m}{b_c}$.

The procedure for identifying the faulty scan channels can be summarized by the following algorithm.

Algorithm 7.4.1 *Non-Adaptive Faulty Scan-Channel Identification:*

1. *Compact the signature $S(x)$ for all m channels in the MUT.*
2. *Generate the reference signature $S^0(x)$,*

$$S^0(x) = \sum_{j=0}^{m-1} S_j^0(x) \ , \tag{7.44}$$

where $S_j^0(x)$ is the computed reference signature for the j^{th} channel. If $S(x) = S^0(x)$, then no faults are detected. Go to [5]. Else, there exists at least one faulty channel. Save $S(x)$ off-line and execute the remaining steps.

3. *Execute* $\mathrm{DTE}_{i,j}$, *for all* $0 \leq i < g_c$, *and* $0 \leq j < b_c - 1$. *During each* $\mathrm{DTE}_{i,j}$, *zeros are shifted-in for all channels not observed by the test experiment. The compacted signatures* $\{S_{i,j}\}$ *are unloaded into the tester at the end of each* $\mathrm{DTE}_{i,j}$ *for off-line analysis.*

4. *When all diagnostic test experiments are executed, apply the off-line* reasoning *procedure:*

   (a) *Compute the complementary signatures* $\{S_{i,b_c-1}\}_{i=0}^{g_c-1}$ *using Equation 7.43.*

   (b) *Calculate the syndrome vector* $\delta =< \delta_{0,0}, \delta_{0,1}, \ldots, \delta_{g_c-1,b_c-1} >$, *where*

   $$\delta_{i,j} = S_{i,j} + S^0_{i,j} \ , \tag{7.45}$$

   and $S^0_{i,j}$ *is the pre-computed reference signature for* $\mathrm{DTE}_{i,j}$.

   (c) *For every* $\delta_{i,j} = 0$, *all the scan channels observed by* $\mathrm{DTE}_{i,j}$ *are error-free (neglecting the controllable impact of traditional signature aliasing). All such channels can be removed from the list of all scan channels. The remaining ones form the List of Candidate Faulty Channels (*L_{CFC}*).*

   (d) *If the cardinality of* L_{CFC} *is less than or equal to* f_c, *then all faulty channels are identified. Go to [5].*

   (e) *If the cardinality of* L_{CFC} *is greater than* f_c, *then either there exist false alarms in* L_{CFC}, *or the MUT contains more faulty scan channels than the assumed* f_c. *If the cardinality of* L_{CFC} *is unacceptably large, three options exist:*

      i. *Take the MUT back to the tester, and execute additional DTEGs, until a satisfactory diagnosis is achieved. This is equivalent to increasing* g_c.

      ii. *If all the possible DTEGs have been exhausted without achieving a satisfactory diagnosis, new test groups can be devised such that the number of scan channels monitored during any DTE is smaller. This is equivalent to increasing* b_c.

      iii. *Follow the non-adaptive diagnosis with an adaptive one, with* L_{CFC} *being the root node in* T_D.

5. *The procedure for identifying the faulty channels is complete.*

The approach in 4(e)ii requires a more complex, programmable CSC, the overhead of which might be unacceptably high. On the other hand, the approach in 4(e)iii requires

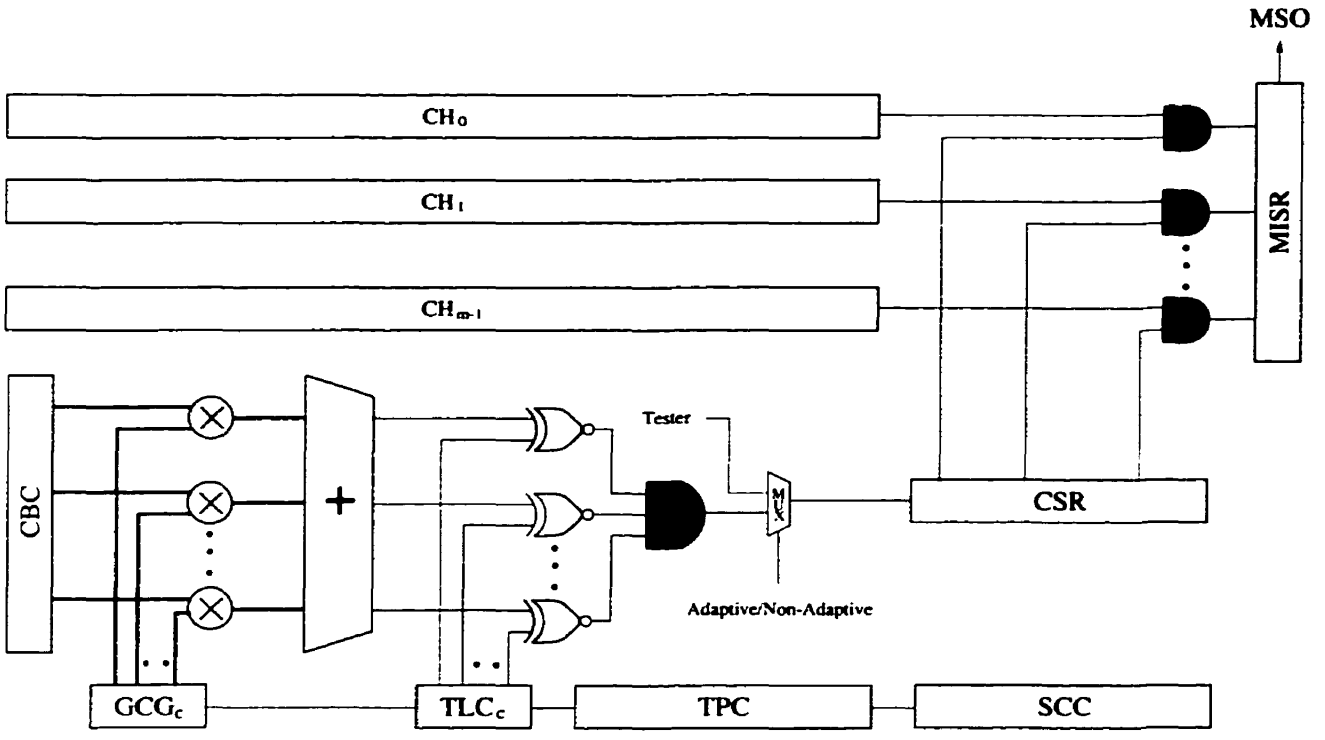


Figure 7.2: Hardware support for non-adaptive faulty-channel identification.

only the addition of a simple multiplexer to accommodate the mixed-mode diagnosis. This is shown in the following section.

7.4.1 Hardware Support for the Diagnostic Algorithm

The diagnostic hardware required to implement the non-adaptive algorithm resides in the channel selection controller. The CSC is used to gate the MISR inputs, such that only the selected subset of channel outputs, dictated by a diagnostic test experiment, are made observable to the compactor.

What follows is a description of the CSC, and the associated hardware components required to support the non-adaptive faulty-channel identification algorithm. The higher the complexity of the MUT, the lower the overhead of the CSC.

The hardware floor that accommodates the algorithm is shown in Figure 7.2.

The CSC consists of the channel select register CSR, and the *channel* Test Group Generator (TGG_c). The CSC also uses signals from the test access port controller TAPC.

All the sequential elements in CSC are implemented with scannable register stages, thus forming a large scan-testable shift register that can be used to check the integrity of the CSC itself.

The CSR, first introduced in Section 6.2.3, is an m-stage register, where each stage is dedicated to driving one of the m 2-input AND gates used to gate the MISR inputs. The CSR is clocked with the TAPC clock $TCLK_2$, in synchronization with the MISR scan-out shift clock.

During the normal operation of the MUT, the CSR is held to the all ones value. In the MSC diagnostic mode, the CSR is loaded with an m-bit code from the TGG_c, and a DTE is executed. The output of CSR represents the m-bit binary label associated with the subset of scan channels involved in the DTE. The CSR contents are held for the whole duration of a DTE, and new values are loaded into these SRSs when the next DTE is about to commence.

The MISR is reseted before the start of signature compaction for every DTE. As mentioned in Chapter 5, the test responses from the channels are fed to the MISR in synchronization with the channels scan-out shift clocks.

The multiplexer at the input of the CSR is added to enable the mixed-mode diagnosis described in Section 7.4. The multiplexer is controlled by the TAPC. If the non-adaptive diagnosis does not yield a satisfactory resolution after exhausting all the possible DTEGs, the multiplexer can be set to load the CSR with adaptive DTEs from the tester.

During diagnosis, system control is passed to the TAPC to execute the test experiments. The TAPC contains the necessary hardware for manipulating the MUT interface to the external world via a standard test bus interface. It also generates the necessary clocking signals $TCLK_1, TCLK_2$ to synchronize between the different components in the CSC on the one hand, and the module's BIST and DFT components on the other hand.

The TAPC supports diagnosis through its $(\log_2 n)$-stage scan cell counter SCC and the $(\log_2 T)$-stage test pattern counter TPC. Both counters are reseted at the beginning of each DTE, and are clocked synchronously with $TCLK_1$.

The SCC is used to control the shifting of test responses captured in the scan cells into the MISR. After n shifts, a single pseudorandom test is completed and a carry-out signal is generated at the output of the counter. The SCC is reset due to the decoded carry-out signal which in turn enables the TPC to increment.

After T tests, a carry-out signal occurs at the output of the TPC indicating the completion of a DTE. Consequently, the signature compacted in the MISR is unloaded into the tester and saved for off-line analysis. A new m-bit word, corresponding to the new DTE, is loaded into the CSR, and the whole process is repeated until the diagnostic test experiments from all groups are executed.

Finally, the TGG_c consists of a Channel Basis Counter (CBC), a *channel* Group Combination Generator (GCG_c), a test group generation circuitry, a *channel* Test experiment Label Counter (TLC_c), and a comparator circuitry.

The CBC is a $(\log_2 m)$-stage counter used to generate the *basis* test experiment groups defined in Section 7.3. It produces the indexes of the channels in base b_c. The CBC is segmented into $\log_{b_c} m$ smaller *basis* counters, each comprising $\log_2 b_c$ stages. The contents of each counter represent one digit of a channel's index in base b_c.

The counters are cascaded such that a carry-out signal at the output of one counter enables the following higher digit counter to increment. All basis counters are clocked synchronously with $TCLK_2$.

The GCG_c generates all the possible distinct g_c group combinations of the basis-DTEGs. It can be implemented as either a special $(\log_2 m)$-stage counter that cycles through the possible group combinations, or as a simple $g_c \times \log_2 m$ EPROM programmed with the g_c groups.

The test group generation circuitry consists of $\log_{b_c} m$ combinational mod $-\ b_c$ multipliers and a $(\log_{b_c} m)$-input mod $-\ b_c$ adder. During each $TCLK_2$ pulse, the contents of the basis counters are multiplied by their corresponding coefficients in the GCG_c, and multiplication results are added. The addition result represents the test experiment label associated with the current test group.

The comparator circuitry compares the generated test experiment label with the contents of TLC_c, and if a match occurs, a 1 is loaded into the i^{th} stage of the CSR, corresponding to the i^{th} scan channel. Otherwise, a zero is loaded into that stage.

The TLC_c is a $(\log_2(b_c - 1))$-stage counter, used to generate the $b_c - 1$ labels associated with each test experiment in a group. As mentioned in Section 7.4, only the first $b_c - 1$ DTEs need to be executed explicitly. Therefore, the label associated with the b_c^{th}-DTE needs not be generated.

The carry-out signal at the output of the TPC causes the TLC_c to increment, so that

a new DTE in the group can be generated. After all $b_c - 1$ test experiments in a group are executed, a carry-out signal at the output of the TLC_c triggers the GCG_c to produce a new group combination. The whole process continues until the required g_c test groups are applied.

If the diagnosis results are unsatisfactory, more test groups can be generated until all possible combinations are exhausted.

The CSC hardware described herein can be easily integrated in different BIST environments due to its flexible implementation.

7.5 Identifying the Faulty Scan Cells

In this section, the working procedure of the non-adaptive algorithm in SSC-mode is explained. It can be used in locating the discrepant scan cells within the identified faulty channel(s). The procedure is an extension of the one used for locating the faulty scan channels.

The diagnosis process is divided into two stages. In the first stage, the non-adaptive algorithm is employed in MSC-mode to identify the faulty scan channels. During the second stage, each faulty channel becomes the subject of the same fault diagnosis procedure, now however, performed in SSC-mode with respect to the scan cells within a channel.

The second stage is repeated for each of the identified faulty channels. For each chain, a number of pre-determined diagnostic test experiment groups (g_{sc}) are executed, each comprising $b_{sc} = f_{sc} + 1$ test experiments, where f_{sc} is the assumed maximum fault multiplicity of the scan cells in any channel.

During each DTE, a single channel is made observable to the MISR. This is done by gating the MISR inputs such that only the channel under diagnosis is made observable, while zeros are forced at the other MISR inputs. Thus, the MISR in this case reduces to an equivalent LFSR. The complete set of T pseudorandom test patterns are applied to the MUT through the channels, and the test responses are shifted out to the compactor. The compacted signature is unloaded into the tester at end of a test experiment, and another DTE is executed.

The DTEs are executed in a sequential manner, and their results are analyzed off-line at the end of the diagnosis process, to determine all the *candidate* faulty scan cells within

a channel.

As the number of scan channels can often be much smaller than the number of scan cells within a channel, the diagnosis time spent in the first stage can be significantly shorter than the second stage.

As explained in Section 7.4, only $b_{sc}-1$ test experiments need to be evaluated explicitly in any test group. The remaining *complementary* one can be obtained using Equation 7.43 with b_c being replaced with b_{sc}. This translates into additional savings in the diagnostic effort equivalent to g_{sc} test experiments, for each of the faulty channels.

In addition to the assumptions made in Section 7.4, the number of scan cells in each channel (n) is assumed to be an integer power of b_{sc}. Thus, the number of cells associated with each DTE in SSC-mode is equal to $\frac{n}{b_{sc}}$.

The procedure for identifying the faulty scan cells in a GSTUMPS module can be summarized by the following algorithm.

Algorithm 7.5.1 *Non-Adaptive Faulty Scan-Cell Identification:*

1. *Execute Algorithm 7.4.1 and identify the faulty scan channels. Maintain their signatures off-line for further analysis.*

2. *For each of the identified faulty scan channels, execute the remaining steps in SSC-mode. If no faults were detected, go to [6].*

3. *Execute* $\mathrm{DTE}_{i,j}$, *for all* $0 \leq i < g_{sc}$, *and* $0 \leq j < b_{sc}-1$. *During each* $\mathrm{DTE}_{i,j}$, *zeros are shifted-in for all scan cells not observed by the test experiment. The compacted signatures* $\{S_{i,j}\}$ *are unloaded into the tester at the end of each* $\mathrm{DTE}_{i,j}$ *for off-line analysis.*

4. *When all diagnostic test experiments are executed, apply the off-line, discrepant scan-cell identification procedure:*

   (a) *Compute the complementary signatures* $\{S_{i,b_{sc}-1}\}_{i=0}^{g_{sc}-1}$ *using Equation 7.43.*

   (b) *Calculate the syndrome vector* $\delta =< \delta_{0,0}, \delta_{0,1}, \ldots, \delta_{g_{sc}-1,b_{sc}-1} >$, *where*

$$\delta_{i,j} = S_{i,j} + S^0_{i,j} \ , \tag{7.46}$$

   and $S^0_{i,j}$ *is the pre-computed reference signature for* $\mathrm{DTE}_{i,j}$.

(c) *For every $\delta_{i,j} = 0$, all the scan cells observed by $\mathrm{DTE}_{i,j}$ are error-free (neglecting the controllable impact of traditional signature aliasing). All such cells can be removed from the list of all scan cells in the faulty channel. The remaining ones form the List of Candidate Faulty Scan Cells (L_{CFSC}).*

(d) *If the cardinality of L_{CFSC} is less than or equal to f_{sc}, then all the faulty cells in the channel are identified. Go to [5].*

(e) *If the cardinality of L_{CFSC} is greater than f_{sc}, then either there exist false alarms in L_{CFSC}, or the channel contains more faulty scan cells than the assumed f_{sc}. If the cardinality of L_{CFSC} is unacceptably large, three options exist:*

i. *Take the MUT back to the tester and execute additional DTEGs, until a satisfactory diagnosis is achieved. This is equivalent to increasing g_{sc}.*

ii. *If all the possible DTEGs for a channel have been exhausted without achieving a satisfactory diagnosis, new test groups can be devised such that the number of scan cells monitored during each DTE is smaller. This is equivalent to increasing b_{sc}.*

iii. *Follow the non-adaptive diagnosis with an adaptive one, with L_{CFSC} being the root node in T_D.*

5. *If there are remaining faulty channels to be diagnosed, go to [2], else, go to [6].*

6. *The procedure for identifying the faulty scan cells is complete.*

The approach in 4(e)ii requires a more complex, programmable CSC, the overhead of which might be unacceptably high. On the other hand, the approach in 4(e)iii requires the addition of a simple multiplexer and some gating logic that accommodates this mixed-mode diagnosis. This can be combined with a mixed-mode scan channel diagnosis, resulting in a more robust, integrated BISD environment. The latter approach is considered in the following section.

7.5.1 Hardware Support for the Diagnostic Algorithm

The channel selection controller supporting the non-adaptive identification of the discrepant scan cells is shown in Figure 7.3.

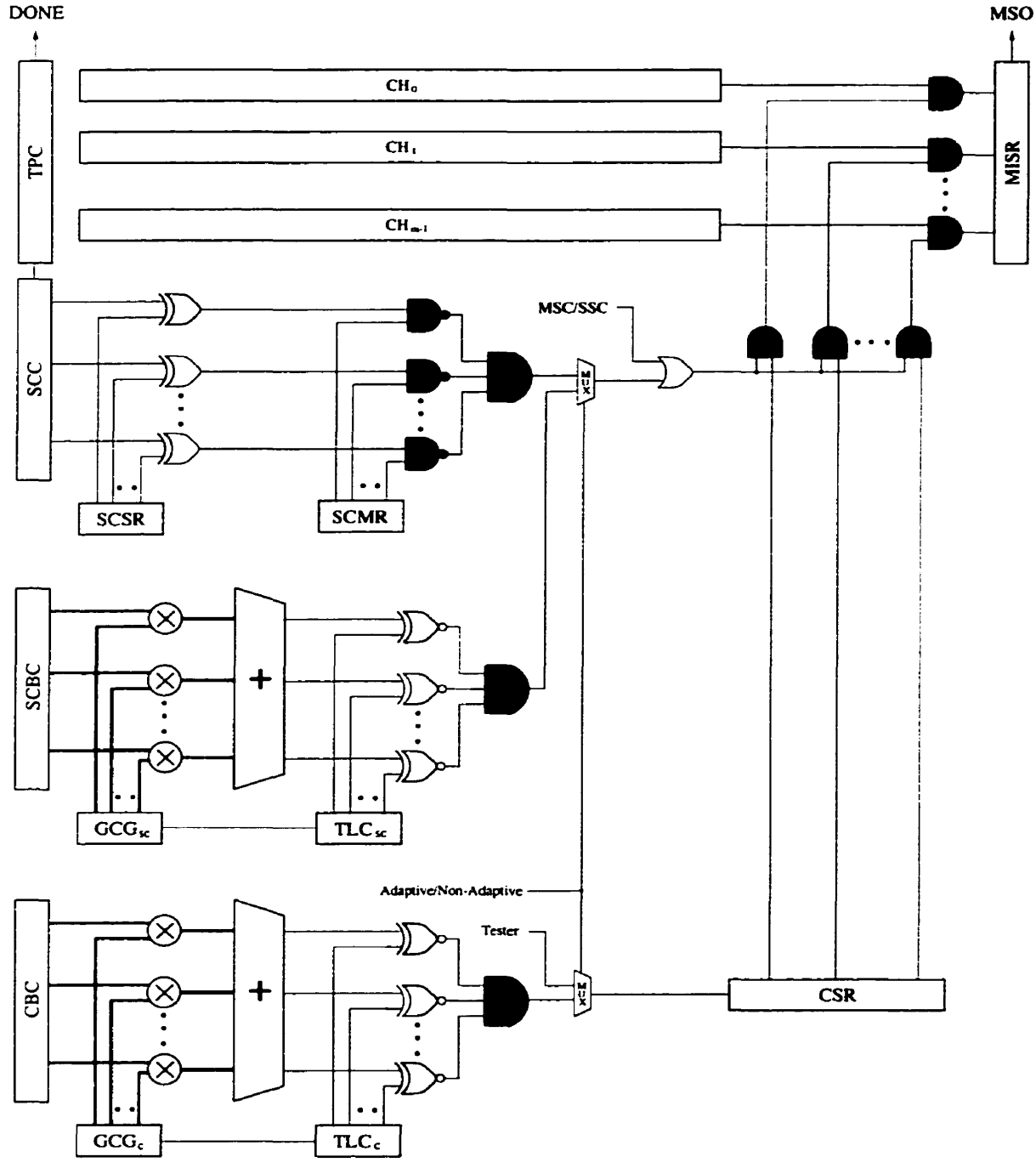


Figure 7.3: Hardware support for non-adaptive faulty scan-cell identification.

The CSC is used to gate the MISR inputs, such that only the selected subset of scan channels/cells, dictated by a DTE, are made observable to the compactor.

The CSC comprises the CSR, and a *scan cell* Test Group Generator (TGG_{sc}). The CSC also uses signals from the TAPC. The SSC diagnostic mode is supported by the scan cell select register SCSR, the scan cell mask register SCMR and their associated compare and mask logic, first introduced in Section 6.3.2.

All sequential elements in the CSC are implemented with scannable register stages, thus forming a large scan-testable shift register that can be used to check the integrity of the CSC itself, as described in Chapter 5.

During the normal operation of the MUT, the CSR is held to the all ones value while the MSC/SSC line is held high. During diagnosis, system control is passed to the TAPC. The TAPC supports the diagnosis process through its SCC and TPC in the same manner described in Section 7.4.1. Both SSC and TPC are reseted at the beginning of each DTE, and are clocked synchronously with $TCLK_1$.

During diagnostic testing, the circuit in Figure 7.3 has two modes of operation when executing the DTEs. The test responses in each DTE are fed to the MISR in synchronization with the channels scan-out shift clocks, and the MISR is reseted before compaction.

In the MSC diagnostic mode, the MSC/SSC line is held high, the CSR is loaded with an m-bit code, and a DTE is executed as described in Section 7.4.1. The multiplexer at the input of the CSR is used to select between the adaptive and non-adaptive test experiments. If the non-adaptive DTEs are used, the CSR is loaded from the *channel* test group generator (TGG_c), otherwise, the DTEs are loaded from the tester.

The CSR is clocked with the TAPC clock $TCLK_2$, in synchronization with the MISR scan-out shift clock. The CSR contents are held for the whole duration of a DTE, and new values are loaded into these SRSs when the next DTE is about to commence.

For the identified faulty channels, the second stage of the diagnostic procedure is executed in SSC-mode. The CSR is loaded with a 1-out-of-m code word corresponding to the faulty channel being diagnosed, and the MSC/SSC line is held low. Thus, the test results from only that channel are fed into the MISR.

During non-adaptive diagnosis, the multiplexer at the input of the OR-gate is set such that the tests from TGG_{sc} can be executed. The TGG_{sc} is similar to the TGG_c described in Section 7.4.1. It consists of a Scan-Cell Basis Counter (SCBC), a *scan cell* Group Com-

bination Generator (GCG_{sc}), a test group generation circuitry, a *scan cell* Test-experiment Label Counter (TLC_{sc}), and a comparator circuitry. The diagnostic procedure is identical to the one described in Section 7.4.1 with m being replaced by n, b_c by b_{sc}, and g_c by g_{sc}. The only difference is that the SCBC is clocked with $TCLK_1$ instead of $TCLK_2$.

If the non-adaptive tests do not yield a satisfactory resolution after exhausting all the possible DTEGs, the multiplexer at the input of the OR-gate is set so that additional adaptive DTEs can be executed. The SCSR and SCMR are loaded with $(\log_2 n)$-bit codes, and the diagnostic procedure described in Section 6.3.2 is applied to identify the error-capturing scan cells. The SCSR and SCMR are also clocked with $TCLK_2$ in synchronization with the MISR shift clock.

The CSC hardware described here can be easily integrated in different BIST environments due to its flexible implementation. The higher the complexity of the MUT, the lower the overhead of the CSC.

7.6 Performance Analysis of the Non-Adaptive Algorithm

The performance of the non-adaptive algorithm is evaluated in this section. In order to quantify the performance, the two metrics of *diagnostic resolution* and *diagnostic efficiency* are defined.

In the following, an analytical performance model is developed that relates the diagnostic resolution and diagnostic efficiency to the number of scan elements, the fault multiplicity, and the number of test experiment groups.

7.6.1 Evaluating the Non-Adaptive Diagnostic Test Resolution

In this section, a new measure of *diagnostic resolution* is introduced. The Diagnostic Resolution (DR) of the non-adaptive algorithm is a measure of the degree of accuracy with which the faulty scan elements can be located after executing g test groups. It is defined by the the ratio of error-free scan elements whose status will be determined as such after

diagnosis. In mathematical terms,

$$DR = \left(1 - \frac{x_{FA}}{x - f}\right) \times 100\% \ , \tag{7.47}$$

where x_{FA} is the number of false alarms after executing g DTEGs, x is the number of scan elements under test, and f is the fault multiplicity.

This metric reflects more accurately the effort that may be required to physically isolate the faults in the MUT with the post-diagnosis analysis described in Chapter 8. The metric also contains enough information to compute the expected number of the *candidate* erroneous scan elements.

Theorem 7.6.1 *The average number of scan elements ($\bar{x}_{FA}$) that will produce false alarms, after executing g diagnostic test experiment groups, is tightly bounded by:*

$$\bar{x}_{FA} < (x - f)\left[1 - \left(1 - \frac{1}{b}\right)^{f}\right]^{g} \ , \tag{7.48}$$

where f is the maximum number of scan elements affected by a fault, b is the number of test experiments in any group, and x is the number of scan elements.

Proof: Using Equation 7.15, the probability that a randomly chosen scan element will be declared error-free after executing a single DTEG is equal to:

$$P(x_{EF_G}) = \left(1 - \frac{1}{b}\right)^{f} \ . \tag{7.49}$$

The probability that a randomly chosen scan element cannot be identified as error-free after executing a single DTEG is equal to $1 - P(x_{EF_G})$. Clearly, such scan elements are either actually faulty with probability $P(x_{E_G})$, or they are false alarms with probability $P(x_{FA_G})$. Since the two probabilities are mutually exclusive, then:

$$\begin{aligned} P(x_{E_G} \cup x_{FA_G}) &= P(x_{E_G}) + P(x_{FA_G}) && (7.50) \\ &= 1 - P(x_{EF_G}) \ . && (7.51) \end{aligned}$$

Thus,

$$P(x_{FA_G}) < 1 - P(x_{EF_G}) \tag{7.52}$$

$$< 1 - \left(1 - \frac{1}{b}\right)^f \quad . \tag{7.53}$$

Since the number of false alarms after executing a single DTEG is on average much greater than the number of scan elements which are actually in error, i.e., $x_{FA_G} >> x_{E_G}$. The above bound is *tight*.

Finally, the probability that a randomly selected scan element will produce a false alarm after g DTEGs is approximately bounded by:

$$P(x_{FA}) < \left[1 - \left(1 - \frac{1}{b}\right)^f\right]^g \quad . \tag{7.54}$$

The simulation results presented in Section 7.7 will show that the above probability remains very close to this estimation.

Hence, the average number of scan elements that will result in false alarms, after executing g diagnostic test experiment groups, is given by:

$$\bar{x}_{FA} = (x - f)P(x_{FA}) \tag{7.55}$$

$$< (x - f)\left[1 - \left(1 - \frac{1}{b}\right)^f\right]^g \quad , \tag{7.56}$$

and the theorem is proved □.

Corollary 7.6.1 *The average number of scan elements ($\bar{x}_{FA,E}$) that will be declared as being either erroneous or false alarms, after executing g diagnostic test experiment groups, is bounded by:*

$$\bar{x}_{FA,E} < (x - f)\left[1 - \left(1 - \frac{1}{b}\right)^f\right]^g + f \quad . \tag{7.57}$$

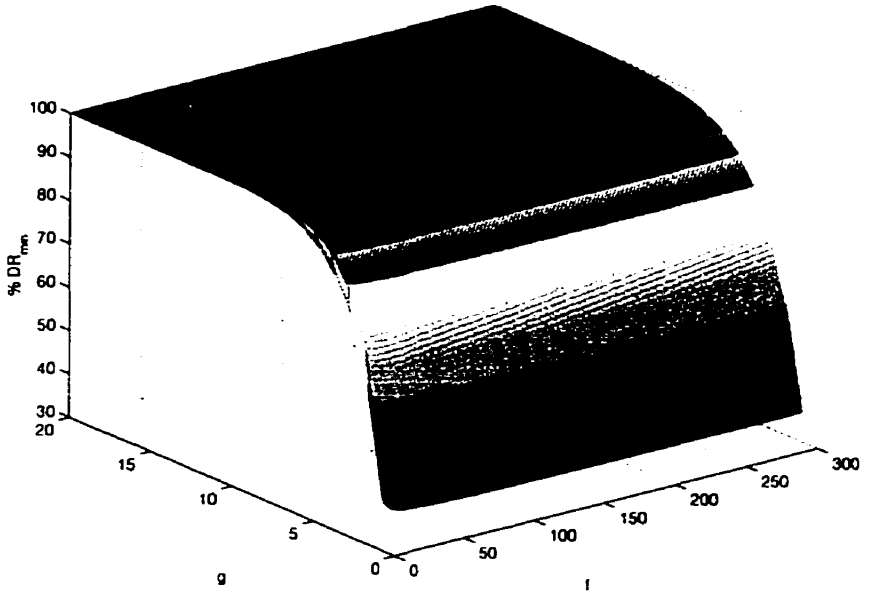


Figure 7.4: The diagnostic resolution as a function of f and g.

Substituting $\bar{x}_{FA}$ from Equation 7.48 into Equation 7.47, the average diagnostic resolution is bounded by:

$$\overline{DR} = \left(1 - \frac{\bar{x}_{FA}}{x - f}\right) \times 100\% \tag{7.58}$$

$$> \left(1 - \left[1 - \left(1 - \frac{1}{b}\right)^{f}\right]^{g}\right) \times 100\% \ . \tag{7.59}$$

For $b = f + 1$,

$$\overline{DR} > \left(1 - \left[1 - \left(\frac{f}{f+1}\right)^{f}\right]^{g}\right) \times 100\% \ . \tag{7.60}$$

A plot of $\overline{DR}_{\min}$ is shown in Figure 7.4 for $b = f + 1$.

Clearly, the diagnostic resolution increases as the number of executed experiment groups increases, for any given fault multiplicity, and decreases as the number of discrepant scan elements increases, for any given number of test groups. The simulation results presented in Section 7.7 strongly confirm these conclusions.

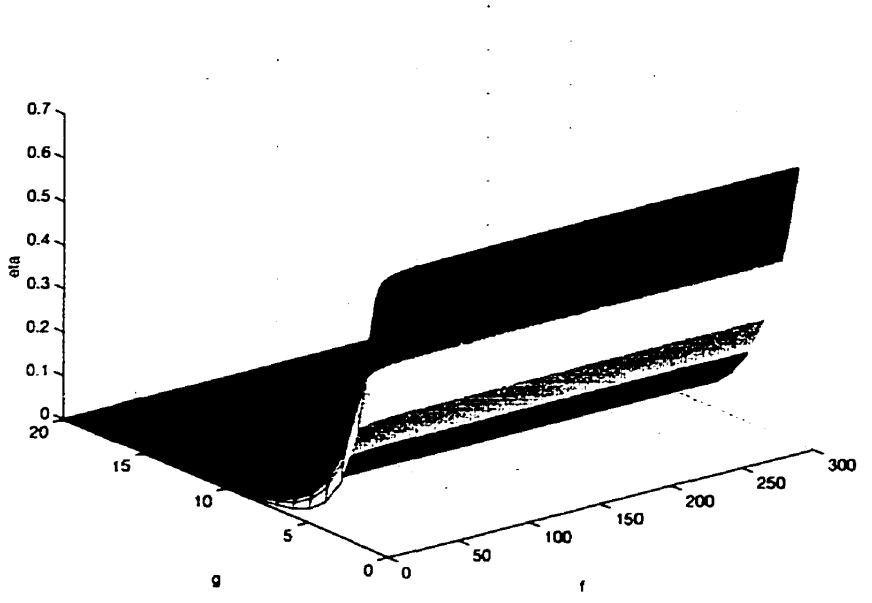


Figure 7.5: The fraction of false alarms as a function of f and g.

7.6.2 Evaluating the Non-Adaptive Diagnostic Test Efficiency

The effectiveness of the non-adaptive scheme is measured by its diagnostic efficiency. This metric, which has been previously used for this purpose in Section 6.5, is based on the number of test experiments required to achieve the desired diagnostic resolution.

The number of DTEGs required to achieve a certain *diagnostic resolution* (η) can be determined from Equation 7.59. Let η be the expected fraction of scan elements that will be declared as false alarms after g DTEGs. Then,

$$\eta = \frac{\bar{x}_{FA}}{x - f} \tag{7.61}$$

$$= \left[1 - \left(1 - \frac{1}{b}\right)^f\right]^g \quad , \tag{7.62}$$

and,

$$g = \frac{\ln \eta}{\ln\left[1 - \left(1 - \frac{1}{b}\right)^f\right]} \quad . \tag{7.63}$$

A plot of η is shown in Figure 7.5 for $b = f + 1$.

Clearly, the fraction of false alarms in the list of candidate erroneous scan elements decreases as the number of executed test groups increases, for any given fault multiplicity.

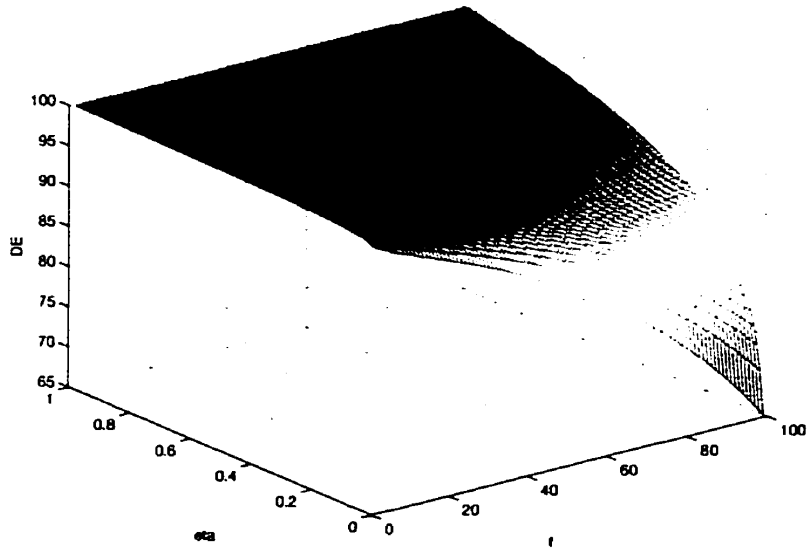


Figure 7.6: The diagnostic efficiency for $x = 3721$, $f \leq 100$, and $b = 61$.

The fraction increases as the number of discrepant scan elements increases, for any given number of experiment groups. The simulation results presented in Section 7.7 strongly confirm these observations.

Finally, the *diagnostic efficiency* of the non-adaptive algorithm can be determined, for a given diagnostic resolution. As stated in Section 6.5, this metric reflects the savings in the diagnostic effort provided by the algorithm as compared to the bottom-line solution for which the number of executed DTEs is always equal to x. Combining Equations 6.40 and 7.63, the diagnostic efficiency is equal to:

$$DE = \left(1 - \frac{N_{\text{DTE}}}{x}\right) \times 100\% \tag{7.64}$$

$$= \left(1 - \frac{g \cdot (b-1)}{x}\right) \times 100\% \tag{7.65}$$

$$= \left(1 - \frac{(b-1)\ln\eta}{x \ln\left[1 - \left(1 - \frac{1}{b}\right)^f\right]}\right) \times 100\% \ . \tag{7.66}$$

For example, for $x = 3721$ scan elements with $f \leq 100$, and $b = 61$, the diagnostic efficiency is shown in Figure 7.6

As can be seen, enormous savings in the diagnostic effort can be obtained while still

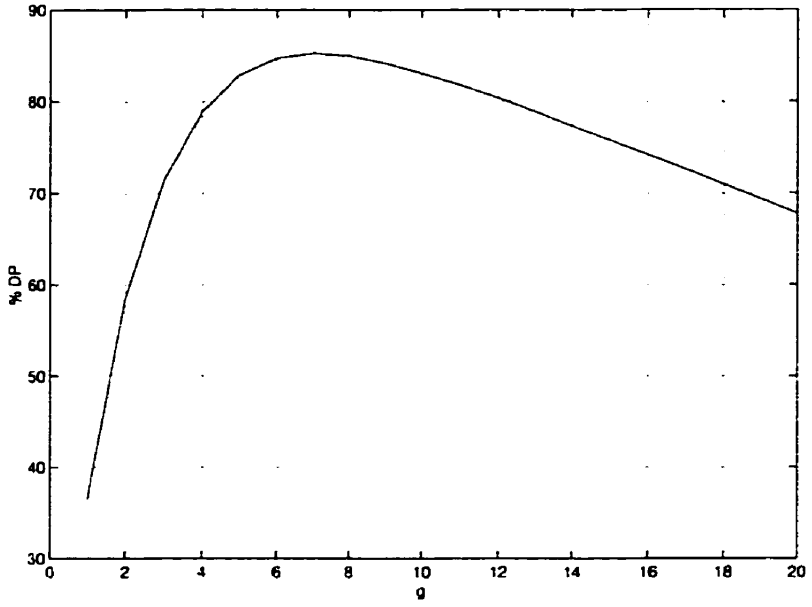


Figure 7.7: The diagnostic performance for $x = 3721$, $f = 60$, and $b = 61$.

achieving a high diagnostic resolution. The simulation results presented in Section 7.7 will show that the above holds for a wide range of fault multiplicities.

Finally, a note about the mixed-mode diagnosis. If the non-adaptive algorithm is intended to be followed by an adaptive one, then the number of DTEGs must be chosen such that it maximizes the *Diagnostic Performance* (DP), where,

$$DP = DR \cdot DE \ . \tag{7.67}$$

A plot of DP for the above example, with $f = 60$, is shown in Figure 7.7.

Clearly, $g = 7$ maximizes the diagnostic performance. Such g results in an optimal combination of diagnostic efficiency and resolution, thus resulting in an optimal performance for the mixed-mode diagnosis.

7.7 Experimental Results

Evaluating the feasibility of the non-adaptive algorithm requires estimating the diagnostic resolution obtained from executing a number of test experiments, and the diagnostic efficiency associated with such a resolution.

To this end, a series of extensive simulation experiments were conducted using different number of scan channels or scan cells within a faulty channel and various fault multiplicities.

The experiments were also used to validate the analytical performance bounds developed in Section 7.6. This section reports the experimental results.

7.7.1 Experimental Setup

For an exact performance evaluation of the algorithm, all possible combinations of faults of multiplicity f that can be chosen from a set of x scan elements, must be simulated.

However, as stated in Section 6.5.1, the complexity of such an approach makes the simulations all but impossible. Consequently, random simulations with *fault sampling* were used instead.

Let j be the number of randomly generated faults for a given multiplicity f. Then for each simulation experiment, the computed diagnostic metrics correspond to an average value taken over j simulated f-tuple faults.

Thus, the average number of false alarms is approximately equal to:

$$\bar{x}_{FA} \approx \frac{\sum_j x_{FAj}}{j} \quad , \tag{7.68}$$

and the average diagnostic resolution is approximated by:

$$\overline{DR} = \left(1 - \frac{\bar{x}_{FA}}{x - f}\right) \times 100\% \quad . \tag{7.69}$$

The diagnostic efficiency does not depend on the number of simulated faults, and is always equal to:

$$DE = \left(1 - \frac{N_{\text{DTE}}}{x}\right) \times 100\% \quad . \tag{7.70}$$

The procedure for evaluating the non-adaptive diagnostic performance was implemented with a C program. The program reads the number of scan elements x, the maximum fault multiplicity f, the number of diagnostic experiments in a test group b, and the number of faults to be randomly generated and evaluated j.

For each fault, the erroneous scan elements were chosen by having a generator of uniformly distributed pseudorandom numbers between 0 and $x - 1$ invoked as many times to obtain an f-tuple with distinct entries.

The random generator used was the Unix© *erand48*(), which returns a pseudorandom value between 0 and 1. This value is multiplied by x and then rounded to the closest integer to obtain the index of the erroneous scan element.

The program measures the average number of resulting false alarms after executing each test group, $\bar{x}_{FA}$, and the associated diagnostic resolutions $\overline{DR}$ and diagnostic efficiency DE. In order to validate the analytical model of the algorithm, the analytical values of $\bar{x}_{FA_{\max}}$ and $\overline{DR}_{\min}$, developed in Section 7.6, were also evaluated for the same data.

All of the reported numbers do not include the extra test experiment, involving all scan elements, required to start the diagnosis process. Furthermore, the number of complementary test experiments is always equal the number of executed test groups, and thus, it is not reported.

The simulation program can be summarized by the following pseudo-code:

Procedure 7.7.1 *Performance of the Non-Adaptive Algorithm:*

1. *Read x, f, b, and j*
2. *Generate the $\lceil \log_b x \rceil$ basis-DTEGs.*
3. *Generate the remaining DTEGs as distinct orthogonal combinations of the basis-DTEGs, for a total of g DTEGs.*
4. *For each number of DTEGs up to g:*
   (a) *Calculate $\bar{x}_{FA_{\max}}$, $\overline{DR}_{\min}$ and DE.*
   (b) *For all j faults to be simulated:*
      i. *Generate a random f-tuple fault.*
      ii. *Calculate the syndrome vector of the j^{th} fault.*
      iii. *Apply the reasoning procedure of Algorithms 7.4.1 and 7.5.1, and obtain the List of Candidate Faulty Scan Elements (L_{CFSE}).*
      iv. *Calculate the number of false alarms:*

$$x_{FA_j} = L_{\text{CFSE}} - f \ . \tag{7.71}$$

(c) Calculate the average number of false alarms:

$$\bar{x}_{FA} = \frac{\sum_j x_{FA,j}}{j} \ . \tag{7.72}$$

(d) Calculate the average diagnostic resolution:

$$\overline{DR} = \left(1 - \frac{\bar{x}_{FA}}{x - f}\right) \times 100\% \ . \tag{7.73}$$

The number of simulated scan elements was chosen between 81 and 10201. These sizes are practical and fit well with the sizes reported in the literature.

For simplicity and exactness of results, the values of x were chosen to be integer powers of b.

The fault multiplicities in our experiments were bounded by $2 \leq f \leq 100$. These numbers were chosen based on the fact that for most circuit nodes, a defect can only affect a limited number of scan cells [38]. The upper bound on f can be easily determined by analyzing the structure of the MUT.

Finally, a maximum of 100000 f-tuple faults where simulated for each fault multiplicity f. It can been shown using probability theory [122] that such a number is sufficient for the simulation sizes above.

7.7.2 Simulation Results

Simulation experiments were conducted on a Sun SPARC-20 station with 256M of RAM, using the above simulation variables.

Tables 7.1- 7.14 contain a summary of these experiments and the comparison between the simulated results and the analytical ones. In each experiment, the number of tests in a DTEG was chosen such that $b = f + 1$. The only exception is the case for $x = 3721$, in which $b = 61$ is used with $f = 10, 30, 60, 90$ in order to evalute the performance of the non-adaptive algorithm when the *actual* fault multiplicity is less or greater than the one for which the DTEGs were generated for. In this case, the DTEGs were generated assuming an $f = 60$.

Both prime and non-prime values were chosen for b. Due to the space limitations, the results for all the possible DTEGs are not tabulated.

g	N_{DTE}	$\bar{x}_{FA}$	$\bar{x}_{FA_{\max}}$	$\%\overline{DR}$	$\%\overline{DR}_{\min}$	$\%DE$
1	2	43.1202	43.8889	45.4175	44.4444	97.5309
2	4	23.1781	24.3827	70.6606	69.1358	95.0617
3	6	12.0220	13.5459	84.7823	82.8532	92.5926
4	8	5.7985	7.5255	92.6601	90.4740	90.1235
5	10	3.2994	4.1808	95.8235	94.7078	87.6543
6	12	1.8067	2.3227	97.7130	97.0600	85.1852
7	14	1.3005	1.2904	98.3538	98.3666	82.7160
8	16	0.4015	0.7169	99.4918	99.0926	80.2470
9	18	0.1006	0.3983	99.8727	99.4959	77.7777
10	20	0.0000	0.2212	100.0000	99.7199	75.3086

Table 7.1: Simulation results for $x = 81$, $f = 2$ and $j = 100000$.

g	N_{DTE}	$\bar{x}_{FA}$	$\bar{x}_{FA_{\max}}$	$\%\overline{DR}$	$\%\overline{DR}_{\min}$	$\%DE$
1	18	209.8032	223.3453	41.5590	37.7868	95.0138
2	36	125.8138	138.9502	64.9544	61.2952	90.0277
3	54	73.7439	86.4453	79.4585	75.9205	85.0415
4	72	42.0943	53.7804	88.2746	85.0194	80.0554
5	90	23.4078	33.4585	93.4797	90.6801	75.0693
6	108	12.5774	20.8156	96.4965	94.2018	70.0831
7	126	6.5375	12.9500	98.1790	96.3927	65.0970
8	144	3.2654	8.0566	99.0904	97.7558	60.1108
9	162	1.5627	5.0123	99.5647	98.6038	55.1247
10	180	0.7035	3.1183	99.8040	99.1314	50.1385
11	198	0.3007	1.9400	99.9162	99.4596	45.1524
12	216	0.1196	1.2069	99.9667	99.6638	40.1662
13	234	0.0432	0.7509	99.9880	99.7908	35.1801
14	252	0.0145	0.4671	99.9960	99.8699	30.1939

Table 7.2: Simulation results for $x = 361$, $f = 18$ and $j = 100000$.

g	N_{DTE}	$\bar{x}_{FA}$	$\bar{x}_{FA_{\max}}$	$\%\overline{DR}$	$\%\overline{DR}_{\min}$	$\%DE$
1	25	402.1176	406.7990	38.2308	37.5117	96.3018
2	50	244.6508	254.2018	62.4192	60.9521	92.6035
3	75	146.6993	158.8465	77.4655	75.5996	88.9053
4	100	87.9230	99.2605	86.4942	84.7526	85.2071
5	125	51.7943	62.0262	92.0439	90.4722	81.5089
6	150	30.0522	38.7591	95.3837	94.0462	77.8107
7	175	17.7209	24.2199	97.2779	96.2796	74.1124
8	200	10.2631	15.1346	98.4235	97.6752	70.4142
9	225	5.8401	9.4574	99.1029	98.5473	66.7160
10	250	3.4920	5.9098	99.4636	99.0922	63.0178
11	275	2.0571	3.6929	99.6840	99.4327	59.3195
12	300	1.4604	2.3076	99.7757	99.6455	55.6213
13	325	0.9021	1.4420	99.8614	99.7785	51.9231
14	350	0.6526	0.9011	99.8997	99.8616	48.2249
15	375	0.4791	0.5631	99.9264	99.9135	44.5266
16	400	0.3092	0.3519	99.9525	99.9460	40.8284

Table 7.3: Simulation results for $x = 676$, $f = 25$ and $j = 100000$.

g	N_{DTE}	$\bar{x}_{FA}$	$\bar{x}_{FA_{\max}}$	$\%\overline{DR}$	$\%\overline{DR}_{\min}$	$\%DE$
1	30	577.4633	582.8740	37.9739	37.3927	96.8782
2	60	353.5140	364.9217	62.0286	60.8033	93.7565
3	90	213.7943	228.4676	77.0361	75.4600	90.6348
4	120	127.5192	143.0374	86.3030	84.6362	87.5130
5	150	75.0306	89.5519	91.9409	90.3811	84.3913
6	180	43.4954	56.0660	95.3281	93.9779	81.2695
7	210	24.8146	35.1014	97.3346	96.2297	78.1478
8	240	13.8986	21.9760	98.5071	97.6395	75.0260
9	270	7.6407	13.7586	99.1793	98.5222	71.9043
10	300	4.1298	8.6139	99.5564	99.0748	68.7825
11	330	2.1904	5.3929	99.7647	99.4207	65.6608
12	360	1.1244	3.3764	99.8792	99.6373	62.5390
13	390	0.5729	2.1139	99.9385	99.7730	59.4173
14	420	0.2788	1.3234	99.9700	99.8579	56.2955
15	450	0.1318	0.8286	99.9858	99.9110	53.1738
16	480	0.0619	0.5187	99.9933	99.9443	50.0520
17	510	0.0272	0.3248	99.9971	99.9651	46.9303

Table 7.4: Simulation results for $x = 961$, $f = 30$ and $j = 100000$.

g	N_{DTE}	$\bar{x}_{FA}$	$\bar{x}_{FA_{\max}}$	$\%\overline{DR}$	$\%\overline{DR}_{\min}$	$\%DE$
1	40	1022.0595	1029.8413	37.7173	37.2431	97.6205
2	80	630.9235	646.2969	61.5525	60.6157	95.2409
3	120	386.1104	405.5961	76.4710	75.2836	92.8614
4	160	233.9969	254.5397	85.7406	84.4887	90.4819
5	200	140.4268	159.7413	91.4426	90.2656	88.1023
6	240	83.3239	100.2488	94.9224	93.8910	85.7228
7	280	48.9710	62.9131	97.0158	96.1662	83.3432
8	320	28.3887	39.4823	98.2700	97.5940	80.9637
9	360	16.3103	24.7779	99.0061	98.4901	78.5842
10	400	9.2132	15.5498	99.4386	99.0524	76.2046
11	440	5.1708	9.7586	99.6849	99.4053	73.8251
12	480	2.8488	6.1242	99.8264	99.6268	71.4456
13	520	1.5396	3.8434	99.9062	99.7658	69.0660
14	560	0.8293	2.4120	99.9495	99.8530	66.6865
15	600	0.4334	1.5137	99.9736	99.9078	64.3070
16	640	0.2232	0.9499	99.9864	99.9421	61.9274
17	680	0.1154	0.5962	99.9930	99.9637	59.5479
18	720	0.0577	0.3741	99.9965	99.9772	57.1684

Table 7.5: Simulation results for $x = 1681$, $f = 40$ and $j = 100000$.

g	N_{DTE}	$\bar{x}_{FA}$	$\bar{x}_{FA_{\max}}$	$\%\overline{DR}$	$\%\overline{DR}_{\min}$	$\%DE$
1	52	1723.5355	1733.0814	37.4851	37.1389	98.1488
2	104	1069.8388	1089.4346	61.1955	60.4848	96.2976
3	156	659.8700	684.8310	76.0657	75.1603	94.4464
4	208	403.7598	430.4925	85.3551	84.3855	92.5952
5	260	245.5830	270.6125	91.0924	90.1845	90.7440
6	312	147.9096	170.1101	94.6351	93.8300	88.8928
7	364	88.5455	106.9331	96.7883	96.1214	87.0417
8	416	52.5440	67.2194	98.0942	97.5619	85.1905
9	468	30.9106	42.2549	98.8788	98.4674	83.3393
10	520	18.0008	26.5619	99.3471	99.0366	81.4881
11	572	10.4355	16.6971	99.6215	99.3944	79.6369
12	624	5.9589	10.4960	99.7839	99.6193	77.7857
13	676	3.3781	6.5979	99.8775	99.7607	75.9345
14	728	1.9028	4.1475	99.9310	99.8496	74.0833
15	780	1.0575	2.6072	99.9616	99.9054	72.2321
16	832	0.5776	1.6389	99.9791	99.9406	70.3809
17	884	0.3171	1.0302	99.9885	99.9626	68.5297
18	936	0.1679	0.6476	99.9939	99.9765	66.6785
19	988	0.0881	0.4071	99.9968	99.9852	64.8273
20	1040	0.0458	0.2559	99.9983	99.9907	62.9761

Table 7.6: Simulation results for $x = 2809$, $f = 52$ and $j = 100000$.

g	N_{DTE}	$\bar{x}_{FA}$	$\bar{x}_{FA_{\max}}$	$\%\overline{DR}$	$\%\overline{DR}_{\min}$	$\%DE$
1	60	557.2219	565.3883	84.9846	84.7645	98.3875
2	120	76.4937	86.1396	97.9387	97.6788	96.7751
3	180	9.4924	13.1238	99.7442	99.6464	95.1626
4	240	1.0506	1.9995	99.9717	99.9461	93.5501
5	300	0.1016	0.3046	99.9973	99.9918	91.9377
6	360	0.0079	0.0464	99.9998	99.9987	90.3252
7	420	0.0005	0.0071	99.9999	99.9998	88.7127
8	480	0.0001	0.0011	99.9999	99.9999	87.1002
9	540	0.0000	0.0002	100.0000	99.9999	85.4878

Table 7.7: Simulation results for $x = 3721$, $f = 10$, $b = 61$ and $j = 100000$.

g	N_{DTE}	$\bar{x}_{FA}$	$\bar{x}_{FA_{\max}}$	$\%\overline{DR}$	$\%\overline{DR}_{\min}$	$\%DE$
1	60	1429.2762	1443.0507	61.2767	60.9035	98.3875
2	120	542.1619	564.1819	85.3112	84.7147	96.7751
3	180	201.3637	220.5752	94.5445	94.0240	95.1626
4	240	73.0922	86.2371	98.0197	97.6636	93.5501
5	300	25.9315	33.7157	99.2974	99.0865	91.9377
6	360	8.9534	13.1816	99.7574	99.6429	90.3252
7	420	3.0081	5.1536	99.9185	99.8604	88.7127
8	480	0.9896	2.0149	99.9732	99.9454	87.1002
9	540	0.3146	0.7877	99.9915	99.9787	85.4878
10	600	0.0961	0.3080	99.9974	99.9917	83.8753
11	660	0.0285	0.1204	99.9992	99.9967	82.2628
12	720	0.0084	0.0471	99.9998	99.9987	80.6504
13	780	0.0024	0.0184	99.9999	99.9995	79.0379
14	840	0.0004	0.0072	99.9999	99.9998	77.4254
15	900	0.0002	0.0028	99.9999	99.9999	75.8130
16	960	0.0001	0.0011	99.9999	99.9999	74.2005
17	1020	0.0000	0.0004	100.0000	99.9999	72.5880

Table 7.8: Simulation results for $x = 3721$, $f = 30$, $b = 61$ and $j = 100000$.

Examining the above tables, the following can be concluded:

1. The average number of false alarms increases with larger sizes of scan elements, and decreases as additional diagnostic test experiments are executed.

2. The average number of false alarms after executing any DTEG increases/decreases with a larger/smaller fault multiplicity than the one for which the DTEGs were generated for.

3. The simulation results confirm very strongly the validity of the developed analytical bounds for estimating the average number of false alarms and the resulting diagnostic resoltuion. As can be seen from Tables 7.7- 7.10, the results hold for fault

g	N_{DTE}	$\bar{x}_{FA}$	$\bar{x}_{FA_{\max}}$	$\%\overline{DR}$	$\%\overline{DR}_{\min}$	$\%DE$
1	60	2291.6226	2303.0473	37.4045	37.0924	98.3875
2	120	1426.2033	1448.7918	61.0433	60.4263	96.7751
3	180	882.6275	911.4002	75.8911	75.1052	95.1626
4	240	542.7735	573.3400	85.1742	84.3393	93.5501
5	300	331.4872	360.6744	90.9455	90.1482	91.9377
6	360	201.2491	226.8916	94.5029	93.8025	90.3252
7	420	121.3088	142.7321	96.6865	96.1013	88.7127
8	480	72.6763	89.7893	98.0149	97.5474	87.1002
9	540	43.1726	56.4843	98.8207	98.4571	85.4878
10	600	25.5044	35.5329	99.3033	99.0294	83.8753
11	660	14.9344	22.3529	99.5921	99.3894	82.2628
12	720	8.6949	14.0617	99.7625	99.6159	80.6504
13	780	5.0136	8.8459	99.8631	99.7584	79.0379
14	840	2.8691	5.5647	99.9216	99.8480	77.4254
15	900	1.6267	3.5006	99.9556	99.9044	75.8130
16	960	0.9146	2.2022	99.9750	99.9399	74.2005
17	1020	0.5092	1.3853	99.9861	99.9622	72.5880
18	1080	0.2782	0.8715	99.9924	99.9762	70.9755
19	1140	0.1525	0.5482	99.9958	99.9850	69.3631
20	1200	0.0820	0.3449	99.9978	99.9906	67.7506

Table 7.9: Simulation results for $x = 3721$, $f = 60$, $b = 61$ and $j = 100000$.

multiplicities which are less than or greater than the one for which the DTEGs were generated for. It should be mentioned that additional simulation experiments with $x = 361, 441, 961, 1681, 2601, 2809$ all confirmed the latter conclusion. These results are not provided here due to the space limitations.

Since the bounds are *tight*, they can be used directly to estimate the number of test groups required to achieve the targeted diagnostic resolution.

4. A high diagnostic resolution is achieved with a small number of test groups over a wide range of fault multiplicities. For example, using a scan channel of length 10201, only 18 out of 102 possible distinct test groups are required to locate 100 faulty scan cells with an average of 1.3201 false alrams.

5. It is clearly evident that the lower bound on the average diagnostic resolution is a strong function of the number of executed test experiment groups. Averaging over all results with $b = f + 1$,

$$\overline{DR}_{\min} \approx (0.3722316)^g \quad . \tag{7.74}$$

This simple formula can be used to quickly estimate the resolution of the non-adaptive algorithm for any number of test groups.

g	N_{DTE}	$\bar{x}_{FA}$	$\bar{x}_{FA_{\max}}$	$\%\overline{DR}$	$\%\overline{DR}_{\min}$	$\%DE$
1	60	2805.5706	2810.7360	22.7328	22.5906	98.3875
2	120	2163.0620	2175.7745	40.4279	40.0778	96.7751
3	180	1664.7134	1684.2544	54.1528	53.6146	95.1626
4	240	1278.0320	1303.7716	64.8022	64.0933	93.5501
5	300	978.5154	1009.2420	73.0511	72.2049	91.9377
6	360	748.4240	781.2484	79.3879	78.4839	90.3252
7	420	570.2290	604.7598	84.2955	83.3445	88.7127
8	480	433.9975	468.1411	88.0474	87.1071	87.1002
9	540	328.8818	362.3853	90.9424	90.0197	85.4878
10	600	249.1969	280.5204	93.1370	92.2743	83.8753
11	660	187.8768	217.1492	94.8258	94.0196	82.2628
12	720	141.7695	168.0939	96.0956	95.3706	80.6504
13	780	106.3749	130.1205	97.0704	96.4164	79.0379
14	840	79.6037	100.7256	97.8077	97.2260	77.4254
15	900	59.6077	77.9711	98.3584	97.8526	75.8130
16	960	44.3407	60.3570	98.7788	98.3377	74.2005
17	1020	32.9717	46.7220	99.0919	98.7132	72.5880
18	1080	24.4223	36.1672	99.3274	99.0039	70.9755
19	1140	18.0098	27.9968	99.5040	99.2290	69.3631
20	1200	13.2509	21.6722	99.6351	99.4031	67.7506
21	1260	9.7497	16.7763	99.7315	99.5380	66.1381
22	1320	7.0995	12.9864	99.8045	99.6423	64.5257
23	1380	5.1988	10.0527	99.8568	99.7231	62.9132
24	1440	3.7823	7.7818	99.8958	99.7857	61.3007
25	1500	2.7334	6.0238	99.9247	99.8341	59.6883
26	1560	1.9694	4.6630	99.9458	99.8716	58.0758
27	1620	1.4148	3.6096	99.9610	99.9006	56.4633
28	1680	1.0156	2.7942	99.9720	99.9230	54.8509
29	1740	0.7263	2.1630	99.9800	99.9404	53.2384
30	1800	0.5125	1.6743	99.9859	99.9539	51.6259
31	1860	0.3681	1.2961	99.9899	99.9643	50.0134
32	1920	0.2560	1.0033	99.9930	99.9724	48.4010
33	1980	0.1807	0.7766	99.9950	99.9786	46.7885

Table 7.10: Simulation results for $x = 3721$, $f = 90$, $b = 61$ and $j = 100000$.

g	N_{DTE}	$\bar{x}_{FA}$	$\bar{x}_{FA_{\max}}$	$\%\overline{DR}$	$\%\overline{DR}_{\min}$	$\%DE$
1	65	2687.4838	2700.3637	37.3693	37.0691	98.5078
2	130	1675.4362	1699.3624	60.9547	60.3971	97.0156
3	195	1038.4794	1069.4236	75.7987	75.0775	95.5234
4	260	643.8658	672.9976	84.9950	84.3161	94.0312
5	325	400.7272	423.5233	90.6612	90.1300	92.5391
6	390	248.7458	266.5269	94.2031	93.7887	91.0468
7	455	157.3656	167.7277	96.3327	96.0912	89.5546
8	520	97.5185	105.5525	97.7274	97.5401	88.0624
9	585	61.1952	66.4251	98.5739	98.4512	86.5702
10	650	40.0313	41.8019	99.0671	99.0258	85.0781
11	715	25.6109	26.3063	99.4032	99.3869	83.5859
12	780	16.4112	16.5548	99.6175	99.6142	82.0937
13	845	10.3435	10.4181	99.7590	99.7572	80.6015
14	910	5.6223	6.5562	99.8690	99.8472	79.1093
15	975	3.3220	4.1259	99.9226	99.9039	77.6171
16	1040	1.8386	2.5964	99.9572	99.9395	76.1249
17	1105	1.0814	1.6340	99.9748	99.9619	74.6327
18	1170	0.9780	1.0283	99.9772	99.9760	73.1405
19	1235	0.3522	0.6471	99.9918	99.9849	71.6483
20	1300	0.1176	0.4072	99.9973	99.9905	70.1561

Table 7.11: Simulation results for $x = 4356$, $f = 65$ and $j = 100000$.

g	N_{DTE}	$\bar{x}_{FA}$	$\bar{x}_{FA_{\max}}$	$\%\overline{DR}$	$\%\overline{DR}_{\min}$	$\%DE$
1	78	3867.3453	3881.3026	37.2490	37.0225	98.7502
2	156	2416.4329	2444.3468	60.7913	60.3384	97.5004
3	234	1501.9885	1539.3882	75.6289	75.0221	96.2506
4	312	929.3168	969.4680	84.9210	84.2695	95.0008
5	390	572.6558	610.5466	90.7082	90.0934	93.7510
6	468	350.9345	384.5069	94.3058	93.7610	92.5012
7	546	214.2553	242.1528	96.5235	96.0709	91.2514
8	624	130.0240	152.5017	97.8903	97.5256	90.0016
9	702	78.5379	96.0418	98.7257	98.4416	88.7518
10	780	47.1675	60.4847	99.2347	99.0186	87.5020
11	858	28.1318	38.0917	99.5435	99.3820	86.2522
12	936	16.7588	23.9892	99.7281	99.6108	85.0024
13	1014	9.8802	15.1078	99.8397	99.7549	83.7526
14	1092	5.8163	9.5145	99.9056	99.8456	82.5028
15	1170	3.3856	5.9920	99.9451	99.9028	81.2530
16	1248	1.9774	3.7736	99.9679	99.9388	80.0032
17	1326	1.1351	2.3765	99.9816	99.9614	78.7534
18	1404	0.6544	1.4967	99.9894	99.9757	77.5036
19	1482	0.3684	0.9426	99.9940	99.9847	76.2538
20	1560	0.2086	0.5936	99.9966	99.9904	75.0040

Table 7.12: Simulation results for $x = 6241$, $f = 78$ and $j = 100000$.

g	N_{DTE}	$\bar{x}_{FA}$	$\bar{x}_{FA_{\max}}$	$\%\overline{DR}$	$\%\overline{DR}_{\min}$	$\%DE$
1	85	4589.9024	4605.6895	37.2192	37.0033	98.8507
2	170	2869.8698	2901.4329	60.7459	60.3141	97.7015
3	255	1787.3829	1827.8073	75.5521	74.9992	96.5522
4	340	1112.9379	1151.4585	84.7772	84.2503	95.4029
5	425	689.7127	725.3810	90.5661	90.0782	94.2537
6	510	426.1137	456.9662	94.1716	93.7496	93.1044
7	595	263.6516	287.8736	96.3938	96.0626	91.9551
8	680	162.5042	181.3509	97.7773	97.5195	90.8058
9	765	99.9081	114.2451	98.6336	98.4374	89.6566
10	850	61.7674	71.9707	99.1551	99.0156	88.5073
11	935	38.0965	45.3392	99.4789	99.3799	87.3580
12	1020	23.3709	28.5622	99.6803	99.6093	86.2088
13	1105	14.5924	17.9932	99.8004	99.7539	85.0595
14	1190	9.0551	11.3351	99.8761	99.8450	83.9102
15	1275	5.6257	7.1408	99.9231	99.9023	82.7610
16	1360	3.5807	4.4985	99.9510	99.9385	81.6117
17	1445	2.2774	2.8339	99.9689	99.9612	80.4624
18	1530	1.4397	1.7853	99.9803	99.9756	79.3131
19	1615	0.9562	1.1247	99.9869	99.9846	78.1639
20	1700	0.6395	0.7085	99.9913	99.9903	77.0146
21	1785	0.4263	0.4463	99.9942	99.9939	75.8653

Table 7.13: Simulation results for $x = 7396$, $f = 85$ and $j = 100000$.

g	N_{DTE}	$\bar{x}_{FA}$	$\bar{x}_{FA_{\max}}$	$\%\overline{DR}$	$\%\overline{DR}_{\min}$	$\%DE$
1	100	6347.3552	6366.5470	37.1611	36.9711	99.0197
2	200	3976.2710	4012.7632	60.6349	60.2736	98.0394
3	300	2480.4419	2529.1997	75.4436	74.9609	97.0591
4	400	1542.8391	1594.1262	84.7259	84.2181	96.0788
5	500	955.5722	1004.7599	90.5398	90.0529	95.0985
6	600	590.5336	633.2889	94.1537	93.7304	94.1182
7	700	362.8770	399.1549	96.4075	96.0484	93.1379
8	800	222.3870	251.5828	97.7984	97.5093	92.1576
9	900	135.6022	158.5699	98.6575	98.4302	91.1773
10	1000	82.5401	99.9448	99.1829	99.0106	90.1970
11	1100	49.9021	62.9941	99.5060	99.3764	89.2167
12	1200	30.1320	39.7045	99.7017	99.6069	88.2365
13	1300	18.0936	25.0253	99.8209	99.7523	87.2562
14	1400	10.8196	15.7732	99.8929	99.8439	86.2759
15	1500	6.4392	9.9416	99.9363	99.9016	85.2956
16	1600	3.8101	6.2661	99.9623	99.9380	84.3153
17	1700	2.2528	3.9495	99.9777	99.9609	83.3350
18	1800	1.3201	2.4893	99.9869	99.9754	82.3547
19	1900	0.7744	1.5690	99.9923	99.9845	81.3744
20	2000	0.4515	0.9889	99.9955	99.9902	80.3941
21	2100	0.2611	0.6233	99.9974	99.9938	79.4138
22	2200	0.1496	0.3929	99.9985	99.9961	78.4335

Table 7.14: Simulation results for $x = 10201$, $f = 100$ and $j = 100000$.

6. A high diagnostic efficiency is achieved with a high resolution over a wide range of fault multiplicities. The percentage reduction in the number of test experiments represents great savings in the diagnostic effort over the straightforward bottom line solution. Indeed, the simulation results indicate that for all values of f that can be considered practical, the scan elements being in error can be correctly identified in a number of BIST sessions, approximately half the number of scan elements under test, and with less than a single false alarm on average.

   For instance, in a 1681-bit scan channel, any 40-tuple of faulty scan cells can be identified with an average of 1.54 false alarms, in a number of test experiments which is only 31% of the total number of scan cells in the channel.

In conclusion, the experimental results strongly support the non-adaptive algorithm as a viable BIST diagnosis technique.

Chapter 8

Locating the Physical Defects

Nowadays, most VLSI circuits are designed at the RTL level. Consequently, the designers of these circuits usually have much better knowledge of the latches and flops than the gates in their designs [38]. Thus, knowing the location(s) of the failing scan cell(s) is very helpful in debugging the MUT.

However, it is still desirable in some cases to identify the actual physical defects, so that a corrective action can be taken. This chapter discusses the possibilities of locating the physical defects after identifying the fault-capturing scan cells embedded in a GSTUMPS environment.

Fault dictionaries and electron probing have been traditionally used in locating the faulty gates and transistors, respectively. Although recent developments have enhanced the capabilities of fault dictionaries and electron probing individually, linking the two techniques in a continuous interaction often yields a finer diagnostic resolution [40].

A quick initial diagnosis of the likely fault sites, using a fault dictionary, eliminates the need to painstakingly probe back from the fault-capturing scan cells, thus reducing the number of required probe points [40]. For VLSI modules whose packaging restricts nodal access, augmenting a guided probe analysis with a fault dictionary is crucial for maintaining observability.

The electron probe on the other hand, can pinpoint the actual physical defect(s) which the dictionary by itself is unable to do. A tight integration of the two in which the fault dictionary advises the probing algorithm on choosing the probe points, can significantly reduce the debug time while achieving a very high diagnostic accuracy [40].

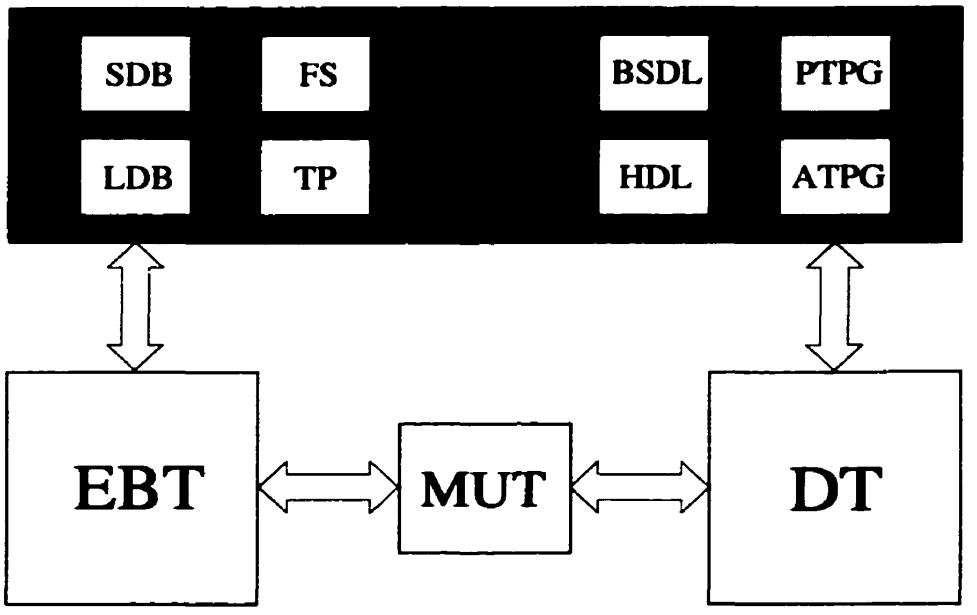


Figure 8.1: A conceptual automatic fault diagnosis system.

The proposed *conceptual* Automatic Fault Diagnosis System (AFDS) is shown in Figure 8.1.

This system can identify in a hierarchical manner the failing boards, MCMs, chips, gates, and their associated I/O pads and interconnects. After identifying the defect(s), a repair call can be generated indicating the necessary corrective action(s).

The diagnosis system consists of a Diagnosis Software Module (DSM), a *simple* Digital Tester (DT), and an Electron Beam Tester (EBT). The communication between the three components is established physically by a local Ethernet and logically by data files having defined interfaces [56].

The DSM controls the complete fault diagnosis process. It includes ATPG tools, Pseudorandom Test Pattern Generators (PTPG), and Fault Simulators (FS) that produce the drive and expect test vectors. It is also responsible for generating and hosting Test Programs (TP) that control how these vectors are used during diagnostic test execution [133].

During test execution, the drive test data is applied to the MUT through its scan channels. The software module compiles the test response data shifted out through the channels and compares it, either on-line or off-line, to the expected response data [22]. The comparison results along with the diagnostic information are written to a file that is fed to a set of diagnostic algorithms.

The MUT must be described in a Hardware Description Language (HDL) such as VERILOG or VHDL. The scan circuitry within a MUT, including boundary scan, must also be specified using, for example, the proposed IEEE Boundary-Scan Description Language (BSDL) [165, 166]. The BSDL describes the types and sizes of the registers that comprise the scan circuitry.

A net-list compiler reads the HDL description and its associated BSDL files. It then generates a Signal Data-Base (SDB) file containing information on how the devices on the MUT are interconnected, how the MUT's scan cells connect to form a scan channel, and which nodes can be driven or sensed by a test program. The SDB information is then used with different test templates to generate the required test and diagnosis test programs.

The computer hosting the software module can be an inexpensive Pentium-based PC or a SUN Sparc station. A standard bus can be used to handle the data transfer between the host computer and the digital tester. The tester on its part contains [6, 22, 55, 161]:

1. A simple micro-controller that coordinates data and controls the signal flow between the host computer and the MUT. The firmware needed for the micro-controller can be stored in an EPROM.
2. A RAM behind each channel, to drive test data and capture test responses.

Basically, the fault diagnosis process can be divided into three steps:

1. Generating the *initial* Candidate Fault List (CFL).
2. A *non-invasive* diagnosis using fault dictionaries to confirm the candidate fault(s) that explain(s) the observed failures.
3. An *invasive* diagnosis using electron probing to conduct a focused inspection of the transistors themselves, thus discovering design errors or physical defects.

Each of these steps will be considered subsequently in following sections.

8.1 Generating the Candidate Fault List

After knowing the exact locations of the fault-capturing scan cells, using either the adaptive or non-adaptive diagnostic algorithms presented in this thesis, the structure of the MUT is analyzed to generate a reduced CFL.

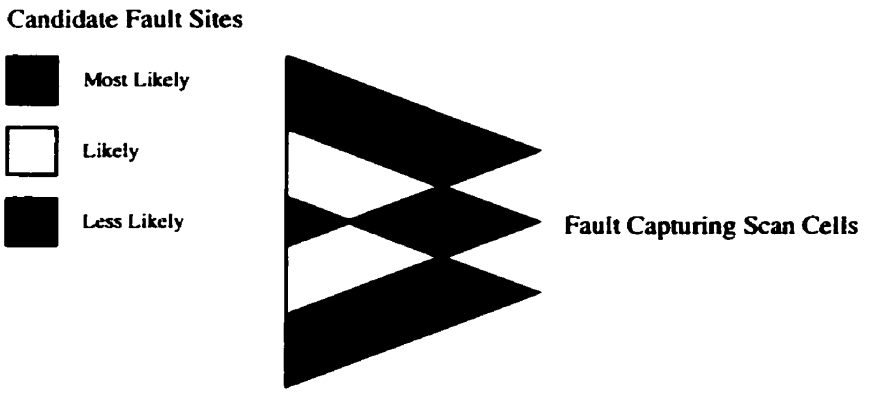


Figure 8.2: Determining the candidate fault list using a backtrace analysis.

This structural analysis tries to find plausible faults that can explain the erroneous behavior of the MUT, and it often yields an acceptable diagnostic resolution [167]. For efficient diagnosis, it is necessary to minimize the size of the initial CFL, thus minimizing the post-test diagnostic-simulation effort. However, the list should be large enough to contain all the faults that could have caused the observed faulty behavior.

In generating the CFL, a reduced model of the MUT is built first. The model consists of the logic *cones* on a *backtrace* starting at each of the discrepant scan cells and terminating at the scan cell boundaries [1, 118]. A cone of logic may overlap with others as shown in Figure 8.2. Thus, the post-test simulation effort is far less than if a complete model of the MUT was used.

In determining the candidate faults, a hierarchy of fault types can be used which include single stuck-at faults, multiple stuck-at faults, memory faults, CMOS faults, shorts and opens [3].

Under the single-site defect assumption [18], there exists a path from the site of the fault to each of the fault-capturing scan cells. Hence, the fault site belongs to the intersection of all logic cones in the reduced model, and the MUT nodes that can exactly affect all the discrepant scan cells are the best candidate faults. These faults are simulated first.

The CFL can be further reduced by taking into account the inversion parities of the paths between the fault site and the capturing scan cells [3].

If no such nodes exist, other MUT nodes, such as the ones that can be propagated to all the faulty scan cells, can be used as candidates. Otherwise, multiple faults are considered and so forth [38].

Using the above criteria, a minimized fault list can be generated to be processed in the next step of diagnosis. Further improvements in generating a CFL can be found in [18, 112].

Finally, it should be mentioned that, in principle, the list of candidate faults can also be determined directly from the faulty signatures compacted at each of the discrepant scan cells. This essentially follows the methods described earlier in Section 3.1.1.

8.2 Dictionary-Based Diagnostics

Once the candidate fault sites and the set of plausible faults are determined by the structural analysis, simulation can be used in postulating a *fault hypothesis*. A fault hypothesis is a set of circuit nodes which are fault candidates explaining the observed erroneous behavior of the MUT during test [56]. This step is limited to the employed fault model, and thus different fault hypothesis must be postulated.

Fault hypotheses can be generated in two different ways. In the first approach, a minimal fault dictionary is constructed first by means of fault simulation. The dictionary database is constructed by injecting all the plausible faults in the CFL to determine the possible responses of the MUT to a given test in the presence of faults. Diagnostic simulation algorithms for sequential circuits has been discussed in [48, 49, 168]. Techniques for simulating faults in the combinational portions of scan circuits have been described in [112, 169, 170].

In order to limit the size of the fault dictionary, without sacrificing the ability to distinguish among different faults, only a meaningful subset of the test-pattern/response data, generated during fault simulation, is selected. In general, only the first failing test that detects each fault is included in the dictionary. Extra tests which *distinguish* between the candidate faults are also included.

Compared to building a complete fault dictionary, this diagnostic fault simulation has the advantage of processing a much smaller set of faults. In addition, since only the fault-capturing scan cells are monitored in a reduced MUT model, the fault simulation effort is greatly simplified.

Using the scan circuitry, the identified diagnostic test patterns are applied to the MUT, and the test responses are unloaded into the tester. All additional processing takes place

off-line in the software module. A fault hypothesis is generated by comparing the actual responses obtained from the faulty MUT, vector by vector, with the pre-computed responses stored in the fault dictionary.

In one pass through the dictionary database, a lookup program in the DSM analyzes each fault syndrome to determine how close a match it provides with the actual MUT response. Faults with the lowest mismatch score represent the best match with the test results.

The output from the lookup program contains a confidence level. This indicates whether the faulty-machine simulation matched all observed passing values in the fault-capturing scan cells, matched all observed failing values, or any combination of the two [40, 118]. The highest confidence level is obtained when both values match, and the lowest confidence is when neither does.

If the test results do not contain enough information to postulate a fault hypothesis, the DSM generates additional test vectors. This process is repeated until a hypothesis is postulated.

The second approach to postulating a fault hypothesis essentially follows the same strategy as with the analytical diagnostic techniques presented in Section 3.2. That is to identify the failing test patterns and then analyze them by post test simulation. The failing tests are fault simulated to determine which of the candidate faults in the CFL can be sensitized by them and observed at the fault-capturing scan cells. The group of faults thus identified constitute the fault hypothesis.

Compared to most existing analytical approaches, the diagnostic algorithms presented in this thesis allow the identification of more failing test patterns. This is because these algorithms identify the fault-capturing scan cells and provide an independent signature for each of them, as opposed to having one signature for all scan cells in a channel. Thus, the sequences observed at each faulty scan cell potentially contain fewer error bits that must be identified by the analytical diagnostic techniques.

Clearly, unless the actual number of failing vectors corresponding to each fault-capturing cell is very small, this second approach to postulating a fault hypothesis is not practical.

Finally, it should be mentioned that the dictionary-based diagnostics maybe the final output of the AFDS in applications where the MUT nodal access is not available for electron-beam probing. In these applications, the dictionary-based diagnostics are extremely valuable.

8.3 Probe-Based Diagnostics

In the past, several probe-based diagnostic systems have been presented [171, 172, 173, 174, 175]. Probing in a scan environment has also been used for some time with modules that employ both internal-scan and boundary-scan techniques [52, 56, 60].

An important component of these systems is the EBT that allows the observation and measurement of signal voltages at internal circuit nodes. This allows for the identification of faulty primitive cells or transistors within the MUT using an *effect-cause* analysis.

In the AFDS under consideration, a *guided-probing* strategy is applied where the DSM is the guide, and the linked EBT represents the probe. The operation of the probe can be automated in the form of a robot arm [1] under the control of the software module. If the operation of the EBT is not automated, a human operator is required to perform the measurement tasks at the coordinates displayed by the DSM.

The selection of the internal MUT nodes to be probed is *automatically* performed by the software module [56]. The required data for this selection includes the SDB, the Layout Data-Base (LDB) of the MUT, and the results of the dictionary-based diagnosis that contain the targeted faults for probing.

During diagnosis, the applied test patterns and initialization sequences are generated *automatically* upon request and *adaptively*, where their generation depends on the predecessing results of the fault diagnosis process [56]. This *online* test generation allows a focused inspection of the MUT.

There are two different starting points for probing. When a *fault hypothesis* is postulated during dictionary-based diagnostics, only nodes which are members of the hypothesis are analyzed. A fault hypothesis corresponds to a rough enclosure of the fault site [56].

Backtrace probing using only nodes from the fault hypothesis allows for a drastic reduction in the number of internal nodes which have to be observed by the EBT and their measurement times [176]. Consequently, the source of the fault can be found in a short time.

In doing so, the probe algorithm in the DSM generates prioritized probe instructions, starting with the most likely candidate node, until a failing node is encountered. As each node is probed, the algorithm maintains a list of nodes that have been probed and found to be good.

When a failing node is encountered, the probe algorithm searches through the SDB and LDB for circuit topology information to determine which inputs to the failing node need to be probed next. It then compares these inputs to the nodes in the fault hypothesis list. If it finds a match, the algorithm directs the next probe to that node. If no match is found, the algorithm makes the node selection based on the topological information alone.

The fault dictionary is referenced to determine the most likely candidate node each time a new list of candidate input nodes is generated for the next probe. This process continues iteratively until a satisfactory diagnosis is obtained.

If either no fault hypothesis was postulated during dictionary diagnostics or all hypotheses are invalidated, probing starts at a fault capturing scan cell. The guided-probing strategy is based on the well known *depth-first* backtracing algorithm [177]. The probe algorithm essentially proceeds in the same manner as above. A fault is located when all predecessors of a faulty node are identified as being fault-free. After resolving one faulty scan cell, the diagnostic system repeats the procedure for the remaining faulty cells.

This type of probing can go along with a lot of measurement nodes, compared to the first one, and therefore it may be very time consuming. However, probing from the faulty scan cells has the advantage of not being restricted to any fault model, since it is only based on signal voltage comparisons. This results in diagnosing multiple classical and non-classical faults [56].

Chapter 9

Conclusion

Testing screens for good modules. Testing digital modules constitutes a major portion of the cost and effort involved in their design, production, and use [1]. The benefits of increased circuit density forces the levels of integration to continue to grow [3]. This in turn exacerbates the testing problem.

BIST has been warmly embraced by the IC industry as a solution for the continuously aggravating testing problem. It shows promise in easing the test application time that hunts VLSI development and its introduction to the manufacturing practice. All these advantages translate into saving dollars.

Testing in a BIST environment has been widely used to provide a simple go/no-go answer. However, when test fall-out is high, it becomes necessary to diagnose faults to improve the yield. Fault diagnosis is important at all levels of integration, ranging from the chip level, MCM level, PCB level, all the way to the system level. The diagnosis process is often *hierarchical* such that the faulty module identified at one level becomes the MUT at the next level [3].

Fault diagnosis is important for a variety of reasons. During the start-up phase of chip production, fault diagnosis can assist in detecting design or process errors. These errors are addressed by improving the quality of the design and the manufacturing process. Later in the product cycle, analysis of failed chips from the field can provide information about their weaknesses. The incidence of physical faults can thus be reduced by designing for reliability with a good quality control in the manufacturing process, and by ensuring proper operating conditions in the field.

An MCM or board-level diagnosis permits faulty chips to be replaced. Finally, fault tolerant systems generally require some diagnostic information in order to allow reconfiguration thus minimizing the system down-time.

Signature analysis is typically used in a BIST environment to compact the outputs of a module into a final signature. Fault detection in this environment is relatively easy compared to fault diagnosis. Valuable diagnostic information is lost as a result of compaction. Consequently, conventional diagnostic techniques can not be directly applied in a BIST environment, and have to be replaced by ones suited for this environment.

In the first part of the thesis, a detailed survey was presented of the diagnostic techniques that can be used in a BIST environment. These SA-based procedures can be employed at different levels of integration, ranging form the chip level all the way to the system level. The applicability, practicality, cost and diagnostic resolution of each method were outlined. The short comings of these methods were analyzed, thus providing the motivation for the thesis.

Then, novel BIST fault diagnosis algorithms were presented for scan-based VLSI modules based on multiple signature analysis. The MUTs were assumed to be designed with GSTUMPS, a generic BISD architecture introduced in this thesis. The development of the GSTUMPS architecture results in highly reliable digital modules. Since faults can be promptly isolated after detection, a recovery process or a corrective action can be executed immediately. No specific fault model has been assumed, and thus all voltage detectable faults are diagnosable.

Diagnosing various malfunctions in a MUT begins by identifying the scan channels/cells that capture errors during test. Based on faulty signature information, the algorithms guarantee the identification of these scan elements, regardless of the number of errors the MUT may produce.

An *adaptive* diagnostic algorithm was developed to minimize the number of signatures that must be collected to determine the fault-capturing scan elements. The adaptive algorithm does not impose any limit on the number of these elements, and can correctly identify any number of them ranging from a single channel/cell to the catastrophic case of all faulty scan channels/cells.

Another *non-adaptive* diagnostic algorithm was also developed. In this algorithm, an upper limit on the number of error-capturing scan elements is assumed. This algorithm also guarantees the identification of all faulty elements. However, depending on the actual

number of faulty channels/cells and the allowed time for diagnosis, the algorithm may also produce some extra false alarms. Several measures were defined to quantify the diagnostic efficiency and resolution of this algorithm.

Both algorithms can be applied at all levels of integration to provide chip level through system level diagnostics. At the system level, the algorithms provide unit level diagnosis. At the PCB or MCM level, they have the ability to isolate the faulty chips. And finally at the chip level, the diagnostic results can be combined with other BIST-based or classical non-BIST techniques to provide a finer gate or even transistor level diagnostic capability.

Furthermore, these diagnostic algorithms minimize the probability of traditional signature aliasing because the test responses are compacted in multiple BIST sessions with different scan channel/cell configurations.

Simulation experiments, assuming various modules having different numbers of scan channels/cells and a range of fault multiplicities, were used to quantify and compare the performance of the above algorithms. The experimental results for the adaptive algorithm clearly showed the enormous savings in diagnosis time in all practical cases, while still ensuring the highest level of diagnostic resolution.

Experimental results for the non-adaptive algorithm showed that a reasonably high diagnostic resolution can be achieved with a small diagnosis time. The results also showed that the number of false alarms produced by the algorithm can be easily controlled at the expense of additional test time. Thus, the non-adaptive algorithm provides the capability for trading-off the diagnostic resolution for diagnosis time.

The cost of these algorithms includes the extra hardware for the channel selection controller, and the extra tester time for collecting multiple signatures in diagnosis mode. This incremental area overhead and the extra tester time maybe a small price to pay for the assured testability, automated diagnosis, and the resultant module quality that ensues.

This project is intended to be a first step in an even larger project. In the future, new approaches to DFD will be explored while considering various possible BIST structures and VLSI architectures. In particular, the extension of our diagnostic methodology to CA-based BIST and ABIST [130, 131] is already under consideration, and the results will be presented in the near future.

Bibliography

[1] Paul H. Bardell, William H. McAnney, and Jacob Savir. *Built-In Test for VLSI: Pseudorandom Techniques.* Wiley-Interscience, New York, 1987.

[2] Laung-Terng (L.-T.) Wang and Paul Y.-F. Wu. Patriot - A Boundary-Scan Test and Diagnosis System. In *Proceedings of the 37th Annual IEEE International Computer Conference*, pages 436–439, 1992.

[3] Miron Abramovici, Melvin A. Breuer, and Arthur D. Friedman. *Digital System Testing and Testable Design.* IEEE Press, 1990.

[4] Janusz Rajski and Jerzy Tyszer. On the Diagnostic Properties of Linear Feedback Shift Registers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10:1316–1322, October 1991.

[5] R. C. Aitken and V. K. Agarwal. A Diagnosis Method Using Pseudo-Random Vectors Without Intermediate signatures. In *Proceedings of the IEEE International conference on Computer-Aided Design*, pages 574–577, 1989.

[6] Mark G. Karpovsky and Saeed M. Chaudhry. Design of Self-Diagnostic Boards by Multiple Signature Analysis. *IEEE Transactions on Computers*, 42(9):1035–1044, September 1993.

[7] N. Kanopoulos, N. Vasanthavada, J.Watterson, and J.J. Hallenbeck. A New Implementation of Signature Analysis for Board Fault Isolation Testing. In *Proceedings of the IEEE International Test Conference*, pages 730–736, 1987.

[8] Mark G. Karpovsky, Tatyana D. Roziner, and Claudio Moraga. Fault Detection in Multiprocessor Systems and Array Processors. *IEEE Transactions on Computers*, 44(3):383–392, March 1995.

[9] Magdy S. Abadir, Ashish R. Parikh, Peter A. Sandborn, Ken Drake, and Linda Bal. Analyzing Multichip Module Testing Strategies. *IEEE Design & Test of Computers*, 11(1):40–51, 1994.

[10] John K. Hagge and Russell J. Wagner. High Yield Assembly of Multichip Modules Through Known Good IC's and Effective Test Strategies. In *Proceedings of the IEEE*, volume 80, pages 1965–1994, December 1992.

[11] E. J. McCluskey. *Logic Design Principles.* Prentice-Hall, Englewood Cliffs, 1986.

[12] Andrew Flint. Test Strategies for a Family of Complex MCMs. In *Proceedings of the IEEE International Test Conference*, pages 436–445, 1994.

[13] Benoit Nadeau-Dostie, Dwayne Burek, and Abu S.M. Hassan. ScanBist: A Multi-frequency Scan-Based BIST Method. In *Proceedings of the IEEE International Test Conference*, pages 506–513, 1992.

[14] K. M. Thompson. Intel and the Myths of Test. In *Proceedings of the IEEE International Test Conference*, page 10, 1995.

[15] Kenneth M. Butler. Deep Submicron: Is Test Up to the Challenge? In *Proceedings of the 26th IEEE International Test Conference*, page 923, 1995.

[16] Craig Hunter. What's So Different About Deep-Submicron Test. In *Proceedings of the 26th IEEE International Test Conference*, page 924, 1995.

[17] Marcelo Lubaszewski, Meryem Marzouki, and Mohamed Hedi Touati. A Pragmatic Test and Diagnosis Methodology for Partially Testable MCMs. In *Proceedings of the IEEE Multi-Chip Module Conference*, pages 108–113, 1994.

[18] John A. Waicukauski and Eric Lindbloom. Failure Dignosis of Structured VLSI. *IEEE Design and Test of Computers*, 6(4):49–60, August 1989.

[19] John P. Hayes. A NAND Model for Fault Diagnosis in Combinational Logic Networks. *IEEE Transactions on Computers*, C-20(12):1496–1506, December 1971.

[20] Ken Kubiak, Steven Parkes, W. Kent Fuchs, and Resve Saleh. Exact Evaluation of Diagnostic Test Resolution. In *Proceedings of the 29th ACM/IEEE Design Automation Conference*, pages 347–352, 1992.

[21] M. Shakeri, K. R. Pattipati, V. Raghavan, F. A. Patterson-Hine, and D. L. Iverson. Multiple Fault Isolation in Redundant Systems. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, volume 2, pages 1795–1800, 1995.

[22] Lee Nayes and Larry Lauenger. Adding Boundary-Scan Test Capability to an Existing Multi-Strategy Tester. In *Proceedings of the IEEE Systems Readiness Technology Conference*, pages 135–142, 1993.

[23] Stephen Y.H. Su and Hede Ma. Designs for Diagnosability and Reliability in VLSI Systems. In *Proceedings of the IEEE International Test Conference: New Frontiers in Testing*, pages 888–897, 1988.

[24] Donald Staab. Practical Issues of Failure Diagnosis and Analysis in a Fast Cycle Time Environment. In *Proceedings of the IEEE International Test Conference: Test and Design Validity*, page 936, October 1996.

[25] David P. Vallett. An Overview of CMOS VLSI Failure Analysis and the Importance of Test and Diagnostics. In *Proceedings of the IEEE International Test Conference: Test and Design Validity*, page 930, October 1996.

[26] D.P. Sieworek and R.S. Swarz. *The Theory and Practice of Reliable System Design.* Digital Press, Bedford Mass, 1982.

[27] S. E. Kreutzer and S. L. Hakimi. System-Level Fault Diagnosis: A Survey. *Micoprocessing and Microprogramming*, 20(4-5):323–330, May 1987.

[28] Dennis Walker and G. S. Hura. A Tutorial on Fault Tolerance Issues with Applications in Distributed Systems. *Microelectronics and Reliability*, 30(6):1029–1037, 1990.

[29] Stephen Y.H. Su, Michal Cutler, and Mingshien Wang. Self-Diagnosis of Failures in VLSI Tree Array Processor. *IEEE Transactions On Computers*, 40(11):1252–1257, November 1991.

[30] Harry Hulvershorn, Paul Soong, and Saman Adham. Linking Diagnostic Software to Hardware Self Test in Telecom Systems. In *Proceedings of the 26th IEEE International Test Conference*, pages 986–993, 1995.

[31] Yoon-Hwa Choi. An Easily-Testable VLSI Array. In *Proceedings of the 2nd International Conference on Supercomputing*, pages 245–249, May 1987.

[32] J. Salinas and F. Lombardi. Diagnosis of Reconfigurable Two-Dimensional Arrays Using a Scan Approach. In *Proceedings of the IEEE International Conference On Wafer Scale Integration*, pages 179–187, 1994.

[33] Kuochen Wang and Wang-Dauh Tseng. Design for Diagnosability and Diagnostic Strategies of WSI Array Architectures. In *Proceedings of the 6th Annual IEEE International Conference On Wafer Scale Integration*, pages 208–217, 1994.

[34] S. Chattopadhyay, D. Roy Chawdhuri, S. Bhattacharjee, and P. Pal Chaudhuri. Board Level Fault Diagnosis Using Cellular Automata Array. In *8th IEEE International Conference on VLSI Design*, pages 343–348, January 1995.

[35] B. R. Wilkins. Hit : A Standard Constructional System for Testability and Maintainability. In *Proceedings of the IEEE International Conference on Computer Design and VLSI in Computers & Processors*, pages 338–341, 1988.

[36] K. Sasidhar, A. Chatterjee, V. K. Agarwal, and J. Hughes. Distributed Probabilistic Diagnosis of MCMs on Large Area Substrates. In *Proceedings of the 26th IEEE International Test Conference*, pages 208–216, 1995.

[37] T. Raju Damarla, Moon J. Chung, Wei Su, and Gerald T. Michael. Faulty Chip Identification in a Multi Chip Module System. In *Proceedings of the 14th VLSI Test Symposium*, pages 254–259, 1996.

[38] Yuejian Wu and Saman Adham. BIST Fault Diagnosis in Scan-based VLSI Environments. In *Proceedings of the IEEE International Test Conference: Test and Design Validity*, pages 48–57, October 1996.

[39] Kenneth M. Butler, Karl Johnson, Jeff Platt, Anjali Jones, and Jayashree Saxena. Integrating Automated Diagnosis into the Testing and Failure Analysis Operations. In *Proceedings of the IEEE International Test Conference: Test and Design Validation*, page 934, October 1996.

[40] Vin Ratford and Paul Keating. Integrating Guided Probe and Fault Dictionary: An Enhanced Diagnostic Approach. In *Proceedings of the International Test Conference : Testings Impact on Design 2 Technology*, pages 304–311, 1986.

[41] Richard E. Anderson, Jerry M. Soden, and Christopher L. Henderson. Future Technology Challenges for Failure Analysis. In *Proceedings of the International Symposium for Tesing and Failure Analysis*, pages 27–32, November 1995.

[42] Jerry M. Soden, Richard E. Anderson, and Christopher L. Henderson. IC Failure Analysis Tools and Techniques - Magic, Mystery and Science. In *Proceedings of the IEEE International Test Conference: Test and Design Validity*, page 935, October 1996.

[43] Colin M. Maunder. *The Board Designer's Guide to Testable Logic Circuits.* Addison-Wesley, 1992.

[44] Colin M. Maunder and Rodham E. Tulloss. *The Test Access Port and Boundary-Scan Architecture.* IEEE Computer Society Press, 1990.

[45] Mark F. Lefebvre. Functional Test and Diagnosis: A Proposed JTAG Sample Mode Scan Tester. In *Proceedings of the IEEE International Test Conference*, pages 294–303, 1990.

[46] David Karpenske and Chris Talbot. Testing and Diagnosis of Multichip Modules. *Solid State Technology*, 34(6):24–26, June 1991.

[47] Cordt W. Starke. Design for testability and diagnosis in a VLSI CMOS System/370 processor. *IBM Journal of Research & Development*, 34(2-3):355–362, March-May 1990.

[48] J. M. Jou and S. C. Chen. A Fast and Memory Efficient Diagnostic Fault Simulation Algorithm for Sequential Circuits. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 723–726, 1994.

[49] S. Venkataraman et al. Rapid Diagnostic Fault Simulation of stuck-at faults in Sequential Circuits Using Compact Lists. In *Proceedings of the 32nd IEEE/ACM Design Automation Conference*, pages 133–138, 1995.

[50] Sreejit Chakravarty. A Sampling Technique for Diagnostic Fault Simulation. In *Proceedings of the 14th IEEE VLSI Test Symposium*, pages 192–197, 1996.

[51] S. Nandi, S. Chattopadhyay, and P. Pal Chaudhuri. Programmable Cellular Automata based Testbed for Fault Diagnosis in VLSI Circuits. In *9th IEEE International Conference on VLSI Design*, pages 61–64, January 1996.

[52] Peter Hansen. Testing Conventional Logic and Memory Clusters Using Boundary Scan Devices as Virtual ATE Channels. In *Proceedings of the 20th IEEE International Test Confernce*, pages 166–173, 1989.

[53] K. E. Posse. A Design-for-Testability Architecture for Multichip Modules. In *Proceedings of the IEEE International Test Conference*, pages 113–121, 1991.

[54] Sandip Kundu. On Diagnosis of Faults in a Scan-Chain. In *Proceedings of the IEEE VLSI Test Symposium*, pages 303–308, April 1993.

[55] Yasuji Oyama, Toshinobu Kanai, and Hironobu Niijima. Scan Design Oriented Test Technique for VLSI's Using ATE. In *Proceedings of the IEEE International Test Conference: Test and Design Validity*, pages 453–460, October 1990.

[56] Matthias Heinitz, Ingo Hollenbeck, Martin Kuboschek, Jan Otterstedt, Christian Sebeke, and Thomas Winkel. Automatic Fault Diagnosis for Scan-based Designs. *Microelectronic Engineering*, 31(1-4):331–338, Feb 1996.

[57] Phillip L. Writer. Design for Testability. In *IEEE Automatic Support System Symposium for Advanced Maintainability*, pages 84–87, October 1975.

[58] Meera M. Pradhan, Rodham E. Tulloss, Harry Bleeker, and Frans P. M. Beenker. Developing a Standard for Boundary Scan Implementaion. In *Proceedings of the IEEE International Conference on Computer Design*, pages 462–466, 1987.

[59] IEEE Standard 1149.3-1993. *IEEE Standard Test Access Port and Boundary-Scan Architecture*. IEEE Standards Board, New York, NY, 1993.

[60] Peter Hansen. Strategies for testing VLSI Boards using boundary scan. *Electronic Engineering*, 61(755):103–111, November 1989.

[61] Gordon R. McLeod. Built-In System Test and Fault Location. In *Proceedings of the IEEE International Test Conference*, pages 291–299, 1994.

[62] K. Totton and S. Shaw. Self-Test: The Solution to the VLSI Test Problems. In *IEE Proceedings - E Computers and Digital Techniques*, volume 135, pages 190–195, July 1988.

[63] Benoit Nadeau-Dostie, Dwayne Burek, and Abu S.M. Hassan. ScanBist: A Multi-frequency Scan-Based BIST Method. *IEEE Design & Test of Comuters*, 11(1):7–17, January 1994.

[64] John C. Chan. An Investigation of the Information Contained in Faulty signatures. *Microelectronics and Reliability*, 32(7):935–940, July 1992.

[65] Charles E. Stroud. Built-In Self-Test for High-Speed Data-Path Circuitry. In *Proceedings of the IEEE International Test Conference*, pages 47–56, 1991.

[66] B. Koenemann, T. Mucha, and G. Zwiehoff. Built-In Logic Block Observation Techniques. In *Proceedings of the IEEE International Test Conference*, pages 37–41, 1979.

[67] A. Krasniewski and S. Pilarski. Circular Self-Test Path: A Low-Cost BIST Technique. In *Proceedings of the 24th IEEE/ACM Design Automation Conference*, pages 407–415, 1987.

[68] M. Ohletz, T. Williams, and J. Mucha. Overhead in Scan and Self-Testing Designs. In *Proceedings of the IEEE International Test Conference*, pages 460–470, 1987.

[69] Louis Y. Ungar. Hierarchical Built-In Test: An Alternative Test and Repair Strategy. In *Proceedings of the IEEE AUTOTESTCON Conference*, pages 456–463, 1995.

[70] A. Pancholy, J. Rajski, and L. J. McNaughton. Empirical Failure Analysis and Validation of Fault Models in CMOS VLSI. In *Proceedings of the IEEE International Test Conference*, pages 938–947, 1990.

[71] Robert W. Bassett, Pamela S. Gillis, and John J. Shushereba. High-Density CMOS Multchip-Module Testing and Diagnosis. In *Proceedings of the IEEE International Test Conference*, pages 530–539, 1991.

[72] W. H. McAnney and J. Savir. There is Information in Faulty Signatures. In *Proceedings of the IEEE International Test Conference*, pages 630–636, 1987.

[73] John P. Robinson and Nirmal R. Saxena. A Unified View of Test Compression Methods. *IEEE Transactions on Computers*, C-36(1):94–99, January 1987.

[74] Solomon W. Golomb. *Shift Register Sequences.* Holden-Day, San Francisco, CA, 1967.

[75] Solomon W. Golomb. *Feedback Shift Registers.* Springer-Verlag, 1984.

[76] W. W. Peterson and E. J. Weldon. *Error Correcting Codes.* MIT Press, Cambridge, MA, 1972.

[77] Paul H. Bardell. Calculating the Effects of Linear Dependencies in m-Sequences Used as Test Stimuli. In *Proceedings of the IEEE International Test Conference*, pages 252–256, 1989.

[78] A. Das and P. Pal Chaudhuri. Efficient Characterization of Cellular Automata. In *IEE Proceedings - E Computers and Digital Techniques*, volume 137, pages 81–87, January 1990.

[79] A. Das and P. Chaudhuri. Vector Space Theoretic Analysis of Additive Cellular Automata and its Application for Pseudoexhaustive Test Pattern Generation. *IEEE Transactions on Computers*, C-42(3):340–352, 1993.

[80] Peter D. McLeod et al. Cellular-Automata-Based Pseudo-Random Number Generators for Built-In Self-Test. *IEEE Transactions on Computer Aided Design of Integrated Circuits & Systems*, 8(8):842–859, August 1989.

[81] Paul H. Bardell. Design Considerations for Parallel Pseudorandom Pattern Generators. *Journal of Electronic Testing: Theory and Applications*, 1:73–87, February 1990.

[82] M. Serra, T. Slater, J. Muzio, and D. Miller. The Analysis of One-Dimensional Linear Cellular Automata and Their Aliasing Properties. *IEEE Transactions on Computer Aided Design*, 9(7):767–778, July 1990.

[83] R. A. Forhwerk. Signature Analysis: A New Digital Field Service Method. *Hewlett-Packard Journal*, 28(9):2–8, May 1977.

[84] S. Lin and D. J. Costello. *Error Control Coding: Fundamentals and Applications.* Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[85] R. David. Testing by feedback shift register. *IEEE Transactions on Computers*, C-29(7):668–673, July 1980.

[86] Bernd Konemann, Joachim Mucha, and Gunther Zwiehoff. Built-In Test for Complex Digital Integrated Circuits. *IEEE Journal of Solid-State Circuits*, SC-15(3):315–319, June 1980.

[87] J. J. LeBlanc. LOCST, a Built-In Self-Test Technique. *IEEE Design & Test of Comuters*, pages 45–52, November 1984.

[88] P. P. Gelsinger. Design and Test of the 80386. *IEEE Design & Test of Comuters*, pages 42–50, June 1987.

[89] Dhiraj K. Pradhan and Sandeep K. Gupta. A Framework for Designing and Analyzing New BIST Techniques and Zero Aliasing Compression. *IEEE Transactions on Computers*, 40(6):743–763, June 1991.

[90] Charles E. Stroud and T. Raju Damarla. Improving the Efficiency of Error Identification via Signature Analysis. In *Proceedings of the 13th IEEE VLSI Test Symposium*, pages 244–249, 1995.

[91] T. Raju Damarla, Charles E. Stroud, and Avinash Sathaye. Multiple Error Detection and Identification via Signature Analysis. *Journal of Electronic Testing: Theory and Applications*, 7:193–207, 1995.

[92] T. Sridhar et al. Analysis and Simulation of Parallel Signature Analyzers. In *Proceedings of the IEEE International Test Conference*, pages 656–661, 1982.

[93] T. W. Williams. *VLSI-Testing.* North-Holland Publishing, Amsterdam, 1986.

[94] Mark G. Karpovsky, Saeed M. Chaudhry, and Lev B. Levitin. Multiple Signature Analysis: A Framework for Built-In Self-Diagnostic. In *Proceedings of the 22nd International Symposium on Fault Tolerant Computing*, pages 112–119, 1992.

[95] André Ivanov and Slawomir Pilarski. Performance of signature analysis: a survey of bounds, exact, and heuristic algorithms. *INTEGRATION, the VLSI journal*, 13:17–38, 1992.

[96] Thomas W. Williams, Wilfried Daehn, Matthias Gruetzner, and Cordt W. Starke. Bounds and Analysis of Aliasing Errors in Linear Feedback Shift Registers. *IEEE Transactions on Computer-Aided Design*, 7(1):75–83, January 1988.

[97] Sandeep K. Gupta and Dhiraj K. Pradhan. A New Framework for Designing & Analyzing BIST Techniques: Computation of Exact Aliasing Probability. In *Proceedings of the IEEE International Test Conference*, pages 329–342, 1988.

[98] F. J. MacWilliams and N. J. Sloane. *Theory of Error Correcting Codes.* North-Holland Publishing, Amsterdam, 1978.

[99] Dhiraj K. Pradhan, Sandeep K. Gupta, and Mark G. Karpovsky. Aliasing Probability for Multiple Input Signature Analyzer. *IEEE Transactions on Computers*, 39(4):586–591, April 1990.

[100] Mark G. Karpovsky, Sandeep K. Gupta, and Dhiraj K. Pradhan. Aliasing and Diagnosis Probability in MISR and STUMPS Using a General Error Model. In *Proceedings of the IEEE International Test Conference*, pages 828–839, 1991.

[101] James E. Smith. Measures of the Effectiveness of Fault Signature Analysis. *IEEE Transactions on Computers*, C-29(6):510–514, June 1980.

[102] T. W. Williams and W. Daehn. Aliasing Errors in Multiple Input Signature Analysis Registers. In *Proceedings of the European Test Conference*, pages 338–345, 1989.

[103] M. Damiani, P. Olivo, M. Favalli, S. Ercolani, and B. Riccó. Aliasing in Signature Analysis Testing with Multiple-Input Shift-Registers. In *Proceedings of the European Test Conference*, pages 346–353, 1989.

[104] André Ivanov and Vinod K. Agarwal. An Analysis of the Probabilistic Behavior of Linear Feedback Signature Registers. *IEEE Transactions on Computer-Aided Design*, 8(10):1074–1088, October 1989.

[105] Janusz Rajski and Jerzy Tyszer. Fault Detection and Diagnosis Based on Signature Analysis. In *Proceedings of the IEEE Interational Symposium on Circuits and Systems*, volume 3, pages 1877–1880, 1991.

[106] K. Wagner, C. K. Chin, and E. J. McCluskey. Pseudorandom testing. *IEEE Transactions on Computers*, C-36:332–343, March 1987.

[107] Kewal K. Saluja and M. Karpovsky. Testing Computer Hardware Through Data Compression in Space and Time. In *Proceedings of the IEEE International Test Conference*, pages 83–88, 1983.

[108] M. Karpovsky and P. Nagvajara. Optimal Time and Space Compression of Test Responses for VLSI Devices. In *Proceedings of the IEEE International Test Conference*, pages 523–529, 1987.

[109] Sudhakar M. Reddy, Kewal K. Saluja, and Mark G. Karpovsky. A Data Compression Technique for Built-In Self-Test. *IEEE Transactions on Computers*, c-37(9):1151–1156, September 1988.

[110] John C. Chan and Baxter F. Womack. On the Relation of Errors and its Syndrome in Signature Analysis. *Microelectronics and Reliability*, 32(10):1379–1383, October 1992.

[111] J. Savir and W. H. McAnney. Identification of Failing Tests With Cycling Registers. *Preceedings of the IEEE International Test Conference*, pages 322–328, 1988.

[112] John A. Waicukauski, Ved P. Gupta, and Sanjay T. Patel. Diagnosis of BIST Failures by PPSFP Simulation. In *Proceedings of the IEEE International Test Conference*, pages 480–484, 1987.

[113] J. J. Curtin and J. A. Waicukauski. Multi-Chip Module Test and Diagnostic Methodology. *IBM Journal of Research and Development*, 27(1):27–34, January 1983.

[114] Y. Arzoumanian and J. Waicukauski. Fault Diagnosis in an LSSD Environment. In *Proceedings of the IEEE Test Conference*, pages 86–88, 1981.

[115] E. B. Eichelberger and E. Lindbloom. Random-Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Test. *IBM Journal of Research and Development*, 27(3):265–272, May 1983.

[116] Jacob Savir. Salvaging Test Windows in BIST Diagnostics. In *Proceedings of the IEEE VLSI Test Symposium*, pages 416–425, April 1997.

[117] Janusz Rajski, Jerzy Tyszer, and Babak Salimi. On the Diagnostic Resolution of Signature Analysis. In *Proceedings of the International Conference on Computer-Aided Design*, pages 364–367, 1990.

[118] Paul H. Bardell and Michael J. Lapointe. Production Experience with Built-In Self-Test. In *Proceedings of the IEEE International Test Conference*, pages 28–33, 1991.

[119] Y.-H. Lee and C. M. Krishna. Optimal Scheduling of Signature Analysis for VLSI Testing. In *Proceedings of the IEEE International Test Conference: New Frontiers in Testing*, pages 443–451, 1988.

[120] Y. Zorian and V. K. Agarwal. A General Scheme to Optimize Error Masking in Built-In Self-Testing. In *Proceedings of the 16th International Symposium on Fault Tolerant Computing*, pages 410–416, 1986.

[121] T. W. Williams, W. Daehn, M Gruetzner, and C. W. Strake. Comparison of aliasing errors for primitive and non-primitive polynomials. In *Proceedings of the IEEE International Test Conference*, pages 282–288, September 1986.

[122] Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, New York, NY, 1991.

[123] J. C. Chan and F. W. Baxter. Diagnostics Based on Faulty Signatures. In *Proceedings of the IEEE International Test Conference*, page 935, 1989.

[124] J. C. Chan and F. W. Baxter. An Algorithm for Diagnostics with Signature Analyzer. In *Proceedings of the IEEE Instrumentation And Measurement Technology Conference*, pages 69–74, 1990.

[125] John C. Chan and Baxter F. Womack. A Study of Faulty Signature for Diagnostics. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, volume 4, pages 2701–2704, New Orleans, LA, MAY 1990.

[126] John C. Chan and Jacob A. Abraham. A Study of Faulty Signatures Using a Matrix Formulation. In *Proceedings of the IEEE International Test Conference*, pages 553–561, October 1990.

[127] T. Raju Damarla and Charles E. Stroud. Design of Signature Registers for Double Error Bit Identification. In *Preceedings of the IEEE International Symposium on Circuits and Systems*, volume 4, pages 65–68, 1996.

[128] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and Their Applications.* Cambridge University Press, Cambridge, 1986.

[129] Mark G. Karpovsky and Prawat Nagvajara. Design of Self-Diagnostic Boards by Signature Analysis. *IEEE Transactions on Industrial Electronics*, 36(2):241–245, May 1989.

[130] Janusz Rajski and Jerzy Tyszer. Test Responses Compaction in Accumulators with Rotate Carry Adders. *IEEE Transactions on Computer Aided Design*, 12(4):531–539, April 1993.

[131] Janusz Rajski and Jerzy Tyszer. Accumulator-Based Compaction of Test Responses. *IEEE Transactions on Computers*, 42(6):643–650, June 1993.

[132] Laung-Terng Wang, Michael Marhoefer, and Edward J. McCluskey. A Self-Test and Self-Diagnosis Architecture for Boards Using Boundary Scans. In *Proceedings of the 1st European Test Conference*, pages 119–126, 1989.

[133] R.E. Tulloss and C.W. Yau. BIST & Boundary-Scan For Board Level Test: Test Program Pseudocode. In *Proceedings of the 1st European Test Conference*, pages 106–111, 1989.

[134] David L. Landis, William A. Check, and Daniel C. Muha. Influence of Built-In Self-Test on the Performance of Fault Tolerant VLSI Multiprocessors. In *Proceedings of the International Conference On Parallel Processing*, pages 114–116, 1987.

[135] John F. Wakerly. *Error Detecting Codes, Self Checking Circuits and Applications.* Elsevier North Holland, New York, NY, 1978.

[136] D. K. Pradhan. A New Class of Error Correcting-Detecting Codes for Fault Tolerant Computer Applications. *IEEE Transactions on Computers*, C-29(6):471–481, June 1980.

[137] Norman Benowitz, Donald F. Calhoun, Gary E. Alderson, John E. Bauer, and Carl T. Joeckel. An Advanced Fault Isolation System for Digital Logic. *IEEE Transactions on Computers*, C-24(5):480–497, May 1975.

[138] Mark G. Karpovsky, Lev B. Levitin, and Feodor S. Vainstein. Diagnosis by Signature Analysis of Test Responses. *IEEE Transactions on Computers*, 43(2):141–152, February 1994.

[139] M. G. Karpovsky, L. B. Levitin, and F. S. Vainstein. Identification of Faulty Processing Elements by Space-Time Compression of Test Responses. In *Proceedings of the IEEE International Test Conference*, pages 638–647, 1990.

[140] H. M. Shao, T. K. Truong, L. J. Deutsch, J. H. Yuen, and I. S. Reed. A VLSI Design of a Pipeline Reed-Solomon Decoder. *IEEE Transactions on Computers*, C-34(5):393–403, May 1985.

[141] M. G. Karpovsky and P. Nagvajara. Board-Level Diagnosis by Signature Analysis. In *Proceedings of the IEEE International Test Conference*, pages 47–53, 1988.

[142] Prawat Nagvajara and Mark G. Karpovsky. Built-In Self-Diagnostic Read-Only-Memories. In *Proceedings of the IEEE International Test Conference*, pages 695–703, 1991.

[143] R. J. McEliece. The Theory of Information and Coding. In *Encyclopedia of Mathematics and Its Applications*, volume 3. Addison-Wesley, Reading, MA, 1977.

[144] R. L. Miller, T. K. Truong, and I. S. Reed. Efficient Program for Decoding the (255,223) Reed-Solomon Code Over GF(2^8) With Both Errors and Erasures, Using Transform Decoding. In *Proceedings of the IEEE*, volume 127, July 1980.

[145] F. J. Garcia-Ugalde. Coding and Decoding Algorithms of Reed-Solomon Codes Executed on a M68000 Microprocessor. In *Proceedings of the* 2^{nd} *International Colloquium*, November 1986.

[146] Paul H. Bardell and William H. McAnney. Self-Testing of Multichip Logic Modules. In *Proceedings of the IEEE International Test Conference*, pages 200–204, 1982.

[147] Sandip Kundu. Diagnosing Scan Chain Faults. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2(4):512–516, December 1994.

[148] Samantha Edirisooriya and Geetani Edirisooriya. Diagnosis of Scan Path Failures. In *Proceedings of the 13th IEEE VLSI Test Syposium*, pages 250–255, 1995.

[149] Geetani Edirisooriya and Samantha Edirisooriya. Scan Chain Fault Diagnosis with Fault Dictionaries. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, volume 3, pages 1912–1915, 1995.

[150] Yiming Gong and Sreejit Chakravarty. On Adaptive Diagnostic Test Generation. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 181–184, 1995.

[151] Donald E. Knuth. *The Art of Computer Programming*, volume 1. Addison-Wesley, Reading, MA, second edition, 1973.

[152] Raymond W. Yeung. On Noiseless Diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(7):1074–1082, July 1994.

[153] Vamsi Boppana, Ismed Hartanto, and W. Kent Fuchs. Full Fault Dictionary Storage Based on Labled Tree Encoding. In *Proceedings of the 14th IEEE VLSI Test Symposium*, pages 174–179, 1996.

[154] H. Y. Chang, E. Manning, and G. Metze. *Fault Diagnosis of Digital Systems.* Wiley Interscience, New York, NY, 1970.

[155] J. Richman and K. R. Bowden. The Modern Fault Dictionary. In *Proceedings of the IEEE International Test Conference*, pages 696–702, 1985.

[156] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* Wiley-Interscience, New York, NY, 1991.

[157] S. Even. *Algorithmic Combinatorics.* The Macmillan Company, 1973.

[158] V. Krishnamurthy. *Combinatorics theory and application.* Ellis Horwoor Limited, 1986.

[159] K. R. Pattipati, M. G. Alexandridis, and J. C. Deckert. A Heuristic Search/Information Theory Approach to Near-Optimal Diagnostic Test Sequencing. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 230–235, 1986.

[160] K. R. Pattipati and M. Dontamsetty. Test Sequencing in Modular Systems. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, volume 3, pages 1221–1223, 1989.

[161] Chi W. Yau and Najmi Jarwala. The Boundary-Scan Master: Target Applications and Functional Requirements. In *Proceedings of the IEEE International Test Conference*, pages 311–315, 1990.

[162] Janusz Rajski and Jerzy Tyszer. Fault Diagnosis In Scan-Based BIST. In *Proceedings of the IEEE International Test Conference*, pages 894–902, 1997.

[163] G. Butter. *Fundamental Algorithms for Permutation Groups.* Spring-Verlag, 1991.

[164] A. Fasster and E. Stiefel. *Group threoretical Methods and their Applications.* Addison Wesley, 1991.

[165] Colin Maunder. Languages to Support Boundary-Scan Test. In *Proceedings of the IEEE International Test Conference*, pages 1104–1105, 1991.

[166] K. P. Parker et al. A Language for Describing Boundary-Scan Devices. In *Proceedings of the IEEE International Test Conference*, pages 222–234, 1990.

[167] F. Hsu, P. Solecky, and R. Beaudoin. Structured Trace Diagnosis for LSSD Board Testing– An Alternative to Full Fault Simulated Diagnosis. In *Proceedings of the 18th IEEE/ACM Design Automation Conference*, pages 891–897, 1981.

[168] E. Rudnick, W. K. Fuchs, and J. H. Patel. Diagnostic Fault Simulation of Sequential Circuits. In *Proceedings of the IEEE International Test Conference*, pages 178–186, 1992.

[169] Junusz Rajski and Henry Cox. A Method of Test Generation and Fault Diagnosis in Very Large Combinational Circuits. In *Proceedings of the IEEE International Test Conference*, pages 932–943, 1987.

[170] Henry Cox and Janusz Rajski. A Method of Fault Analysis for Test Generation and Fault Diagnosis. *IEEE Transactions on Computer Aided Design*, 7(7):813–833, 1988.

[171] N. Yamaguchi, T. Sakamoto, H. Nishioka, T. Majima, T. Satou, H. Shinada, H. Todokoro, and O. Yamada. E-Beam Fault Diagnosis System for Logic VLSIs. *Microelectronic Engineering*, 16:121–128, 1992.

[172] Hiromu Fujioka and Koji Nakamae. LSI Failure Analysis with CAD-Linked Electron Beam Test System and Its Cost Evaluation. *IEICE Transactions on Electronics*, E77-C:535–545, April 1994.

[173] M. Melgara, M. Battu, P. Garino, J. Dowe, Y. J. Vernay, M. Marzouki, and F. Boland. Automatic Location of IC Design Errors Using An E-Beam System. In *Proceedings of the IEEE International Test Conference*, pages 898–907, 1988.

[174] H. Nijima, Y. Tokunaga, S. Koshizuka, and K. Yakuwa. Electron Beam Tester Integrated into a VLSI Tester. In *Proceedings of the IEEE International Test Conference*, pages 908–913, 1988.

[175] Koji Nakamae, Katsuyoshi Miura, and Hiromu Fujioka. VLSI Testing with CAD-Linked Electron Beam Test System. *Microelectronic Engineering*, 31(1-4):319–330, Feb 1996.

[176] N. Kuji and K. Shirakawa. Cone/Block Methods for Logic Simulation Time Reduction in E-Beam Guided-Probe Diagnosis. *IEICE Transactions on Electronics*, E77-C:560–566, April 1994.

[177] W. Groves. Rapid Digital Fault Isolation with Fastrace. *Hewlett-Packard Journal*, 30:8–12, March 1979.