The Dissertation Committee for Krishna Vijaya Chakravadhanula certifies that this is the approved version of the following dissertation:

New Test Vector Compression Techniques

Based on Linear Expansion

Committee:

_____
Nur Touba, Supervisor

_____
Anthony P. Ambler

_____
Ross Baldick

_____
Craig Chase

_____
Doug Burger

New Test Vector Compression Techniques Based on Linear Expansion

by

Krishna Vijaya Chakravadhanula, B.E., M.S.

Dissertation

Presented to the Faculty of the Graduate School of
the University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2004

To My Beloved Family

Acknowledgments

I am grateful to Prof. Nur Touba for his guidance, encouragement and help at each and every step of my research. While working with him I have always admired his clarity in thought and exposition and hope to achieve the same high standards that he has set for himself. It has been a great experience working with him.

I would like to thank Profs. Anthony Ambler, Ross Baldick, Craig Chase and Doug Burger for serving in the dissertation committee and reviewing this dissertation.

I thank all the members of my research lab for their friendship and help during my life as a graduate student. I am also thankful to Melanie, Linda, and Shirley for their administrative support.

During my stay in Austin I have been fortunate enough to make some great friends. I am thankful to all my friends for making my stay in UT a truly memorable one. I would like to mention just a few by name: Sandip, Arindam, Hari, Abhijit, Shovan, Souvik, Manoj, Kartik, Bala, Sreangsu, Dwip, Sudipta, Sayantan, Jayanta, Vikas, Jayabrata, Aditya, Sugato, Shalini, and Arnab. The Friday night parties with great food, the cricket tournaments, and the animated discussions that we used to have, were all a lot of fun.

During my summer internships I have had the opportunity to meet several people who have influenced me positively. Johnny LeBlanc, Adam Cron, Vishwani Agrawal, and Rishi Chawla are among the many that not only provided me with tremendous support but also helped broaden my horizons.

Finally, I thank my parents, brother, sister, and the rest of my family for their continuous love and support in all my endeavors. I also appreciate the hard work that they had to put in to provide me with all the opportunities that I needed. I dedicate this dissertation to them.

May 2004

New Test Vector Compression Techniques Based on Linear Expansion

Publication No. _____

Krishna Vijaya Chakravadhanula, Ph.D.

The University of Texas at Austin, 2004

Supervisor: Nur A. Touba

This dissertation considers the problem of reducing the storage as well as the bandwidth (data transfer rate between tester and chip) requirements of automatic test equipment (i.e., testers). Several new test vector compression schemes based on *linear expansion* are presented. The compressed test vectors are stored on the tester and transferred to the chip where a linear expansion network is used to decompress them. The lossless compression techniques described here significantly reduce the test data stored on the tester compared with conventional external testing, but do not require the hardware overhead and complexity of full stand-alone built-in self-test (BIST). One of the contributions of this dissertation is efficient compression techniques that do not require any constraints on the automatic test pattern generation (ATPG) process, thus simplifying the design flow. A new form of linear feedback shift register (LFSR) reseeding is described which allows *partial dynamic* reseeding and achieves greater encoding efficiencies than previous forms of LFSR reseeding. A new hybrid BIST scheme is proposed that uses an "incrementally guided LFSR" to provide very attractive tradeoffs between test length and tester storage requirements while using very simple

on-chip hardware. A technique is described that combines LFSR reseeding and statistical coding in a powerful way by taking advantage of the large solution space of linear equations to find LFSR seeds that can be efficiently encoded using a statistical code. A new scheme for combinational linear expansion is proposed that uses *adjustable width* expansion to eliminate the need that every scan bit-slice be in the output space of the linear decompressor, while providing greater compression than fixed width expansion techniques. Finally, a compression scheme is described that combines three different stages of linear expansion to achieve extremely high encoding efficiencies while requiring low hardware overhead as it configures the decompressor out of the scan cells themselves. Both the adjustable width and 3-stage decompression schemes provide the nice feature that any scan vector can be efficiently compressed regardless of the number or distribution of specified bits, thus allowing the use of any ATPG procedure without any constraints.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

To ensure high reliability of the finished product, manufacturing test incurs significant costs. Each second of test application time for which the chip sits in the socket of the expensive automatic test equipment (ATE) adds significantly to the test cost. Testers (ATE) have limited speed, memory and I/O channels. If the amount of test data (i.e., test vectors and output response for the chip) is greater than the tester's memory, then memory reload(s) may be required which adds significantly to the test time. The test data bandwidth is the rate at which data can be transferred between the tester and the chip. The test data bandwidth depends on the tester's speed and number of I/O channels (as illustrated in Fig. 1.1). The test application time can be no less than the total amount of test data divided by the available test data bandwidth.

The current trend in the semiconductor industry is to move towards very deep sub-micron technologies to increase functionality and achieve higher performance. The amount of logic on the chip has thus increased manyfold. Consequently the amount of test data volume required for testing such large designs is also growing rapidly. The limited test data bandwidth between the tester and the chip is becoming a major bottleneck that is expected to become much worse as the projections in [Khoche 00] indicate. Conventional external testing where all test data is stored on the tester (ATE) and transferred to/from the circuit-under-test (CUT) is thus becoming increasingly difficult. There is a need for new *test resource partitioning* schemes where some hardware is added on the chip to ease the burden on the external tester by reducing tester storage requirements and test data bandwidth requirements by an order of magnitude or more. The focus of this dissertation is to develop techniques that reduce the tester storage requirements and tester bandwidth requirements.

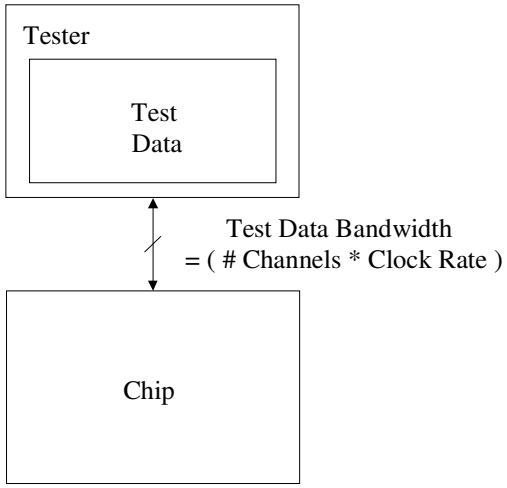


Figure 1.1. Block Diagram Illustrating Conventional External Testing

One well-known approach is to use built-in self-test (BIST). The idea behind BIST is to transfer onto the chip the major functionalities previously carried out by the external test equipment. BIST involves performing test pattern generation and output response compaction on the chip. One of the common BIST architectures is the IBM pioneered scan-based STUMPS architecture [Bardell 82] shown in Figure 1.2. In this architecture, the linear feedback shift register (LFSR) is used to generate the pseudo-random patterns, and the output response of the circuit-under-test is compacted in a multiple input signature register (MISR). The most economical BIST schemes are based on pseudo-random pattern testing. Pseudo-random pattern generation is performed by using linear feedback shift registers (LFSRs) or cellular automata. Generally it is not possible to get high fault coverage with pseudo-random BIST due to the presence of random-pattern-resistant faults within the circuit-under-test [Eichelberger 83]. One approach for solving this problem is to modify the circuit-under-test to eliminate the random pattern resistance by inserting test points. Because this involves modifying the functional logic, it can lead to degradation in the system performance. Another approach is to modify the pattern generator using additional hardware to generate patterns that detect these random-pattern-resistant faults [Touba 96b], [Kiefer 98, 00], [Fagot 98]. But the use of this additional hardware requires additional silicon area. Another issue with pseudo-

random BIST is excessive power dissipation due to the high amount of switching activity that it causes which may cause thermal problems.

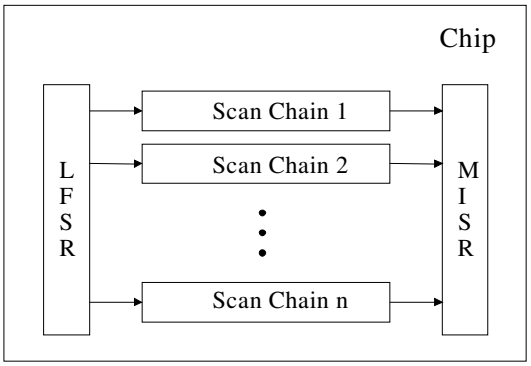


Figure 1.2. Block Diagram Illustrating the STUMPS Architecture

Many techniques have been developed to improve the fault coverage of pseudo-random BIST by adding additional hardware. These techniques include inserting test points in the CUT to make it random-pattern-testable [Eichelberger 83], [Touba 96a], adding weight logic [Brglez 89], [Muradali 90], [Pomeranz 93] or mapping logic [Chatterjee 95], [Touba 95a, 95b] to bias the patterns towards those that detect the hard faults, and adding logic to embed deterministic patterns in the pseudo-random sequence [Touba 96b], [Kiefer 98]. The major drawback of most of these techniques is that the additional hardware that is required does not scale well for increasingly large circuits, and they generally involve a lot of additional control complexity requiring a complex BIST controller with a lot of routing of control signals. If the purpose of adding the BIST circuitry is to provide self-test capability out in the field where there is no access to a tester, then a stand-alone BIST scheme is necessary. However, if the purpose of adding the BIST circuitry is to reduce test data storage and bandwidth requirements for the tester during production test, then a stand-alone BIST scheme is overkill.

A lot of work has also been done in the area of directly compressing the test data. Compressing the output response is relatively easy because lossy compression techniques can be employed (e.g., using a multiple input signature register). However, compressing test vectors is much more difficult because lossless compression techniques must be used.

Recently, as reducing test vector volume has become such an important problem, a lot of research has been done on lossless compression techniques for test vectors. These techniques include using run-length codes [Jas 98], statistical codes [Jas 99], Golomb codes [Chandra 00], frequency directed codes [Chandra 01], VIHC codes [Gonciari 02], LZ77 [Wolff 02], Mutation codes [Reda 02], packet-based codes [Khoche 02], [Volkerink 02], and non-linear combinational codes [Reddy 02], [Li 03]. These techniques involve storing the test vectors in a compressed form on the tester, and then transferring them to the chip where they are decompressed using on-chip hardware. Complex compression techniques cannot be used since the decompression phase would then involve a significant hardware overhead.

A third major class of test vector compression schemes is based on linear expansion. In these techniques, the data on the tester is decompressed on the chip by performing only linear operations. This includes techniques using linear feedback shift registers (LFSRs) and combinational linear expansion circuits like XOR gates. Some commercial tools for compressing test vectors - including TestKompress from Mentor Graphics [Rajski 02] and SmartBIST from IBM/Cadence [Könemann 01] - are based on linear expansion circuits. One of the nice features of linear expansion techniques is that they reduce the test data stored on the tester compared with full external testing, but do not require the hardware overhead and complexity of full stand-alone BIST. Linear expansion circuits exploit the unspecified (don't care) bit positions in test cubes (deterministic test vectors in which bits unassigned by automatic test pattern generation, ATPG, are left as don't cares) to achieve large amounts of compression. Linear expansion circuits are also called linear decompressors. Chapter 2 describes the idea of linear expansion in greater detail and also classifies the existing linear expansion techniques.

The contribution of this dissertation is new test vector compression schemes that are based on linear expansion. Innovative techniques including partial reseeding, seed compression, adjustable width expansion and 3-stage decompression allow the bandwidth and storage requirements for the external tester to be reduced by an order of magnitude or more, while using low hardware overhead. One of the contributions of this dissertation is efficient techniques that provide high compression without the need for any constraints

on the ATPG process. This allows the designer the flexibility of using any ATPG procedure to generate the test vectors that can then be compressed using the compression schemes described here. Thus the proposed techniques can be easily integrated into an existing test flow. Some of the techniques described can be used not only for compressing a fully deterministic test set but for mixed-mode BIST (only the vectors for the random-pattern-resistant faults are compressed) as well, which can help to achieve higher compression. If some test scheduling is done then some of these techniques are also well suited for testing core-based system-on-chip (SOC) designs.

The compression technique described in chapter 4 is a new form of LFSR reseeding that is *dynamic* (i.e., the seed is incrementally modified while test generation proceeds) and allows *partial* reseeding (i.e. $n<r$ bits can be used for r-bit LFSR). In addition to using a simpler hardware implementation, the proposed scheme achieves greater compression than previous forms of LFSR reseeding which have been *static* (i.e., test generation is stopped to load the seed) and have required *full* reseeding (i.e., $n=r$ bits are used). Chapter 5 describes a new hybrid BIST scheme that uses a generalized form of partial dynamic reseeding to provide very attractive tradeoffs between test time and tester storage requirements while using very simple on-chip hardware. The data on the tester is used to incrementally guide the on-chip LFSR so that it can embed patterns that detect the random-pattern-resistant faults in the pseudo-random sequence of the LFSR. A lossless compression technique is described in chapter 6 that combines LFSR reseeding and statistical coding in a powerful way by taking advantage of the large solution space of linear equations to find LFSR seeds that can be efficiently encoded using statistical codes.

Chapter 7 describes a new scheme for combinational linear expansion that uses the ability to adjust the width of the linear expansion each clock cycle to eliminate the need that every scan bit-slice be in the output space of the linear decompressor. Moreover, it allows greater compression to be achieved than fixed width expansion techniques since the ratio of the number of scan chains to the number of tester channels can be scaled much larger. Finally, the compression scheme described in chapter 8 combines three different stages of linear expansion to achieve extremely high encoding efficiencies while

requiring low hardware overhead as it configures the decompressor out of the scan cells themselves. Both the adjustable width and 3-stage decompression schemes provide the nice feature that any scan vector can be efficiently compressed regardless of the number or distribution of specified bits, thus allowing the use of any ATPG procedure without any constraints.

Chapter 9 summarizes all the work in this dissertation and discusses possible topics for future research.

Chapter 2

Linear Expansion

Test vector compression techniques based on the idea of linear expansion have become very popular recently. While these schemes are simple to implement, the amount of compression achieved can be an order of magnitude or more when compared to conventional external testing. Thus these techniques help to alleviate the tester storage problem by significantly reducing the amount of test data that needs to be stored on the tester. These techniques also significantly reduce the test data bandwidth requirements by requiring fewer channels between the tester and the chip. Another nice feature of these techniques is that the functional logic does not need to be modified in any way in order to improve the fault coverage. This is because these techniques can perform lossless compression of deterministic test vectors which detect all the testable faults within the circuit. Thus the fault coverage provided by these schemes is identical to that of external testing.

In techniques based on linear expansion, the data on the tester is decompressed on the chip by performing only linear operations. Since only linear operations are used for the decompression process, this greatly simplifies the task of computing the compressed data. Thus, simple techniques like Gauss-Jordan Elimination [Cullen 97] can be used during the computation process, instead of the more complex techniques required for solving non-linear systems. This is one of the significant advantages of linear expansion. Since only linear operations are allowed, the hardware that is used to implement these schemes typically comprises of linear feedback shift registers (LFSRs) and exclusive-or (XOR) gates.

Test cubes (deterministic test vectors in which bits unassigned by automatic test pattern generation are left as don't cares) typically have only 1-10% of the bits specified, while the rest are don't cares. Linear expansion schemes work well because they only encode the specified bit information within the test cubes while ignoring the don't care bits. Since the number of specified bits is small, there is much less information to encode

as compared to encoding the entire test cube. Linear expansion techniques exploit this to achieve high compression.

The test methodology for schemes using linear expansion typically comprises of two phases. In the first phase, standard test pattern generation tools can be used to generate the deterministic test cubes that detect the targeted faults within the circuit. The bits unassigned by the ATPG tool are left as don't cares instead of filling them with random values. These test cubes are then encoded based on the operation of the linear expansion scheme that is being used. The compressed data is then stored on the tester instead of the test cubes themselves. In the second phase, the compressed data is sent to the chip where it is decompressed using the linear expansion circuit. The output of the linear circuit then feeds the scan chains within the circuit.

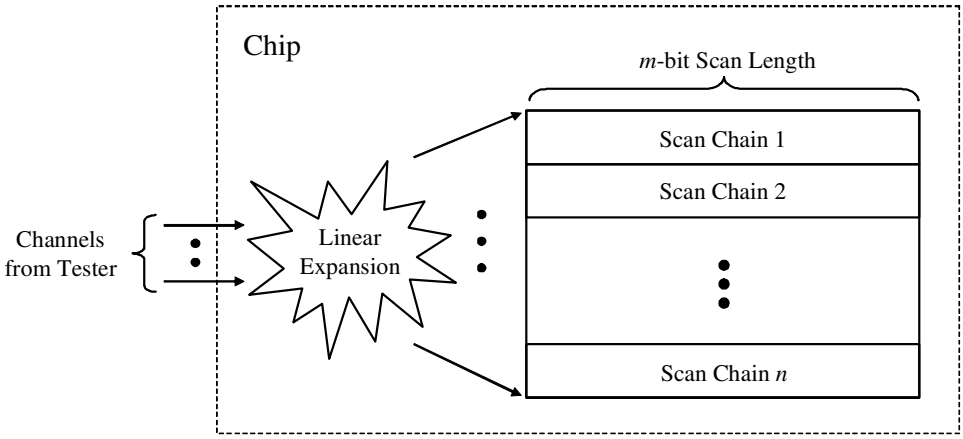


Figure 2.1. Basic Architecture for Linear Expansion

Figure 2.1 shows the basic architecture for a linear expansion scheme. The inputs to the linear expansion network come from the tester while its outputs feed the *n* scan chains within the circuit. The channels from the tester send the compressed data to the linear expansion network on the chip which then decompresses it and loads it into the multiple scan chains. Note that the number of channels from the tester is typically much smaller than the number of scan chains, which helps to reduce the test data bandwidth requirements. In external testing, the number of scan chains is limited by the number of

channels from the tester. But linear expansion techniques allow a much higher number of scan chains, which reduces the length of each scan chain, thereby reducing the time required to shift a vector into the scan chains.

Since the data on the tester is decompressed on the chip by performing only linear operations, each decompressed bit can be represented as a linear (modulo-2) sum of the compressed bits. This idea was originally studied by Könemann for use in LFSR-reseeding [Könemann 91] (described in chapter 3). If each bit stored on the tester is represented by a "free-variable" that can be assigned any value (0 or 1), then the basic idea with linear expansion techniques is to send the data from the tester through a linear expansion circuit (which can be combinational or sequential) as it is shifted into the scan chains. The contents of each scan cell can then be represented by a linear equation in terms of the free-variables on the tester. Test cubes typically have only a few number of the bits specified, while the rest are don't cares. Thus, a system of linear equations can be formed consisting of one equation for each specified bit position in a test cube. The solution to this system of linear equations gives an assignment of values to the free-variables on the tester that will produce the test cube in the scan chains (details about this process are described in chapter 4 as well as in [Könemann 91] and [Hellebrand 95a]). If the number of free-variables used for a test cube is sufficiently larger than the number of specified bits in the test cube, then the probability of not being able to solve the system of linear equations becomes negligible. For LFSR reseeding, it has been shown that if the number of free-variables is 20 more than the number of specified bits, then the probability of not being able to solve the system of linear equations is less than 10^{-6} [Könemann 91]. Thus, linear expansion schemes are very good at exploiting the don't cares in the test cubes to provide high compression.

One of the metrics that is used to evaluate these schemes is *encoding efficiency*, which is defined as the ratio of the number of specified bits in the test cube to the number of bits used to encode the test cube.

$$\textit{encoding efficiency} = \frac{\text{Number of specified bits}}{\text{Number of bits stored on tester}}$$

So an encoding efficiency of 1 corresponds to the case where the test cubes were encoded with the same number of bits as the total number of specified bits in all the test cubes. The goal of linear expansion techniques is to achieve an encoding efficiency as close to 1 as possible. It is unlikely that an encoding efficiency greater than 1 can be achieved using a scheme based purely on linear expansion since that would mean being able to solve a system of linear equations that has more equations than variables.

2.1 Classifying Linear Expansion Techniques

A number of test data compression schemes based on linear expansion have been proposed. They can be categorized based on two attributes. One is whether they receive data from the tester in a continuous-flow (i.e., "streaming" data) or whether they are periodically-loaded (i.e., require some delay between loads), and the other is whether they use a fixed number of free-variables per test cube or whether they vary the number of free-variables used per test cube.

Continuous-flow schemes can be directly connected to the tester and operate very efficiently because they simply receive the data as fast as the tester can transfer it. The linear expansion network keeps receiving data from the tester every clock cycle while concurrently shifting data into the scan chains. Another nice feature of continuous-flow techniques is that the test data can be applied in a similar manner to conventional ATPG test vectors. From a tools integration standpoint, this is very nice since it mimics the standard behavior of normal scan chains. Even though the linear expansion network may feed a large number of scan chains within the chip, to the tester it appears as if there are only as many scan chains in the chip as there are channels connecting the tester to the chip.

Periodically-loaded schemes typically can achieve slightly higher encoding efficiency, but they require some test scheduling to fully utilize the tester bandwidth during the time period between loads. They are thus well suited for a system-on-chip (SOC) environment where multiple cores are being tested concurrently and the data from the tester can be scheduled to be delivered to each core as needed so that the tester

bandwidth is never wasted. Once the requisite test data has been sent to a core, the expansion network within the core begins decompressing the test data and shifts it into the scan chains. While the first core is decompressing the test data, the tester can send data to another core. Some test scheduling thus needs to be done in order to fully utilize the tester resources. Since most periodically-loaded schemes are based on reseeding of LFSRs, chapter 3 describes the idea of LFSR reseeding in greater detail.

Schemes that use a fixed number of free-variables per test cube have less control complexity. Each test cube is treated the same, so no extra control information is needed per test cube. The drawback is that in order to encode all test cubes, the number of free-variables used per test cube must be sufficient to encode the most specified test cube (i.e., the worst-case). If s_{max} is the number of specified bits in the most specified test cube, then the number of free-variables used to encode each and every test cube is determined by s_{max}. If s_{avg} is the average number of specified bits per test cube, then if there is a significant difference between s_{max} and s_{avg}, the encoding efficiency will be substantially degraded. To avoid this, constraints on the ATPG process (especially on static and dynamic compaction) are needed to keep s_{max} from becoming too large. This may result in more test vectors than would be generated if the ATPG was not constrained. Schemes that can vary the number of free-variables used for each test cube avoid the need for constraints on the ATPG process. These schemes have high encoding efficiency regardless of s_{max} and s_{avg}. They do not require any special ATPG tool support, and may require fewer test vectors since more compaction can be done during ATPG. The drawback of these schemes is the need for additional control bits for each test cube which may offset some of the gains, as well as some extra hardware complexity.

Some of the existing linear expansion schemes – including the schemes described in this dissertation - are now categorized based on the above two attributes.

Periodically-loaded schemes with a fixed number of free-variables per test cube:

The earliest linear expansion schemes were periodically-loaded schemes that used a fixed number of free-variables per test cube. These schemes used full reseeding of

LFSRs [Könemann 91] or multiple-polynomial LFSRs [Hellebrand 95a]. More recent work uses partial LFSR reseeding [Krishna 01] (described in chapter 4).

Periodically-loaded schemes with a variable number of free-variables per test cube:

The schemes described in [Zacharia 95, 96], [Rajski 98a] varies the number of free-variables per test cube by reseeding variable-length LFSRs. The scheme in [Hellebrand 95b] tries to encode multiple test cubes per seed. The scheme in [Krishna 02] (described in chapter 6) compresses LFSR seeds using a statistical code, so the number of bits stored on the tester for each test cube varies depending on the amount of seed compression achieved. The scheme in [Krishna 03a] (described in chapter 5) embeds deterministic test cubes within a sequence of pseudo-random test vectors such that each test cube requires a variable number of bits from the tester before it can be embedded in a suitable test vector.

Continuous-flow schemes with a fixed number of free-variables per test cube:

The scheme described in [Jas 00] is a continuous-flow scheme that internally uses LFSR reseeding. Its encoding efficiency is limited by the fact that it uses small LFSRs and directly feeds one scan sub-chain. The schemes in [Bayraktaroglu 01, 03] and [Mitra 03] use a combinational linear expansion circuit to decompress one "bit-slice" (see Fig. 2.2) of a set of multiple scan chains at a time. The encoding efficiency is limited by the fact that they use a fixed number of free-variables per bit-slice. The features of these schemes are described in greater detail in chapter 7. The scheme described in [Rajski 02] injects free-variables from the tester into a ring generator [Mrugalski 03] (which is similar in nature to an LFSR) which then passes through a phase shifter and into the scan chains. This method is tightly coupled with the ATPG software to ensure that each test cube that is generated can be encoded. This scheme is used in the TestKompress tool from Mentor Graphics and has been shown to achieve excellent compression for industrial circuits. One other scheme that technically can be classified as a linear expansion scheme is serial/parallel scan chains (Illinois scan architecture) [Hamzaoglu 99], [Hsu 01]. Here the linear expansion circuit does not have any XOR gates, but

simply routes the same scan-in line from the tester to multiple scan chains. If there are conflicting specified values, then the test vector is applied in serial mode with no compression.

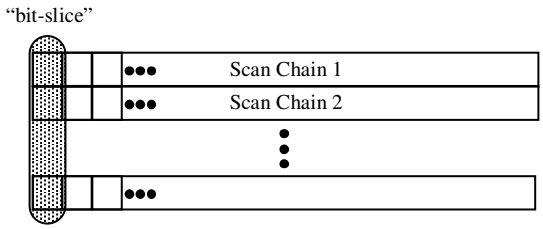


Figure 2.2. Definition of a "Bit-Slice" of Multiple Scan Chains

Continuous-flow schemes with a variable number of free-variables per test cube:

The first continuous flow scheme that used a variable number of free-variables per test cube was part of the SmartBIST family of decoders described in [Könemann 01]. In this method one channel from the tester is used for controlling a clock gating mechanism that controls when the scan chains are loaded from a set of LFSRs receiving the free-variables from the tester. When a sufficient number of free-variables have been injected into the LFSRs to solve the linear equations for the specified bits in the next bit-slice of the scan chains, then the clock gating mechanism allows the scan chains to be loaded. This scheme is capable of applying any test cube regardless of the number of specified bits in it. A scheme based on static LFSR reseeding was described in [Volkerink 03] which uses the same control mechanism as [Könemann 01], but uses a large LFSR (e.g., 500 bits) along with a shadow register to allow it to decompress more than one bit-slice at a time. A scheme based on seed overlapping was described in [Rao 03] which is based on the method in [Bayraktaroglu 01] but allows seeds of different sizes to be used. It also decompresses one scan bit-slice at a time and requires a special constrained ATPG. The scheme in [Krishna 04] is a continuous-flow scheme with a variable number of free-variables per test cube which combines three different stages of linear expansion. This scheme is described in greater detail in chapter 8. A continuous-flow scheme using an adjustable width combinational expansion circuit [Krishna 03b] is described in chapter 7.

All the techniques described in this dissertation can also be classified based on the above two attributes. Chapters 4, 5, and 6 describe periodically-loaded techniques while chapters 7 and 8 describe two continuous-flow techniques. Since the periodically-loaded techniques are based on the idea of LFSR reseeding, the next chapter provides an overview of the LFSR reseeding technique followed by a summary of the previous work in this area.

Chapter 3

Overview of LFSR Reseeding

One attractive approach that has been developed for compressing the amount of test data that needs to be stored on the tester and transferred to the chip is to use linear feedback shift register (LFSR) reseeding [Könemann 91]. This approach is illustrated in Figure 3.1. Most techniques based on the idea of LFSR reseeding fall under the category of periodically-loaded techniques. Since some of the compression schemes described later on in this dissertation are based on the idea of LFSR reseeding, this chapter gives an overview of the LFSR reseeding approach.

An *LFSR seed* is the starting state of an LFSR when the LFSR is run in autonomous mode to fill a set of scan chains with a test vector (if there are m bits in each scan chain, then the LFSR is run for exactly m cycles to fill the scan chains). Different LFSR seeds will produce different test vectors. Given a set of deterministic test cubes (test vectors in which bits unassigned by ATPG are left as don't cares), the idea in LFSR reseeding is to compute a set of seeds that when expanded by the LFSR will produce the deterministic test cubes. A seed can be computed for each test cube by solving sets of linear equations based on the feedback polynomial of the LFSR [Könemann 91]. So instead of storing each full test vector on the tester, a much smaller LFSR seed is stored instead (in Figure 3.1, a set of L test cubes are stored on the tester as a set of L seeds). The set of seeds stored on the tester are transferred to the LFSR one at a time and expanded into the corresponding full test vector in the scan chains.

Since the seeds are much smaller than the full test vectors, the test data storage and bandwidth requirements for the external tester can be reduced by an order of magnitude or more. Another nice property of LFSR reseeding is that it can be seamlessly combined with pseudo-random built-in self-test (BIST) to form a mixed-mode testing approach. The LFSR can first be run in an autonomous mode to generate some number of pseudo-random patterns to detect the random-pattern-testable faults, and then reseeding can be used to generate deterministic test cubes to detect the random-pattern-resistant faults.

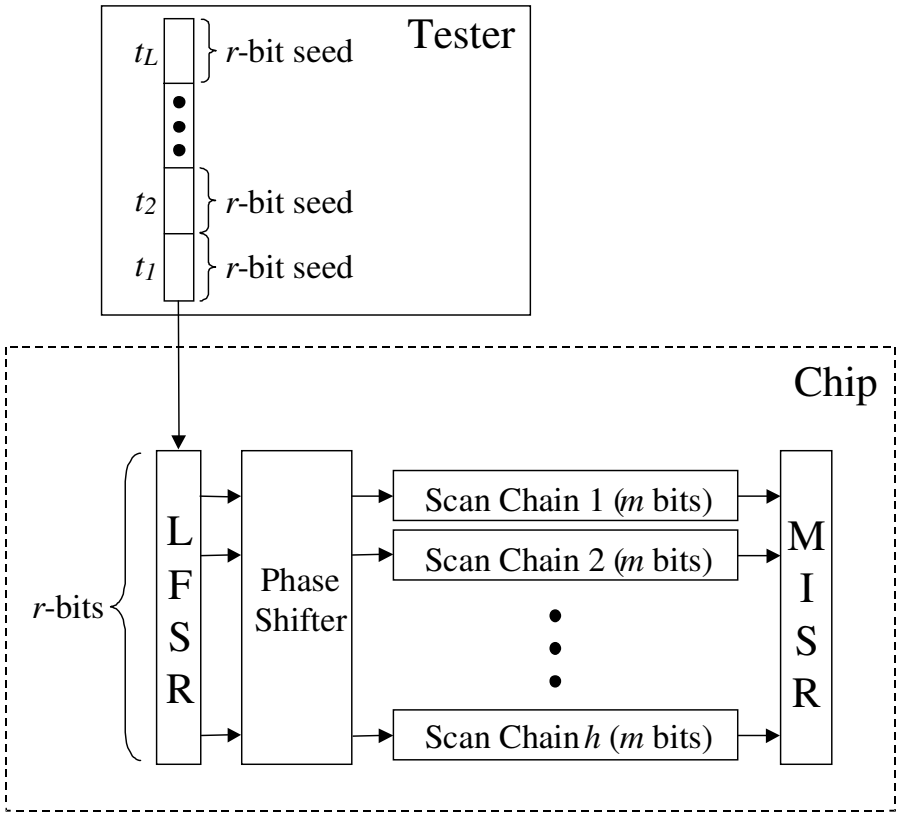


Figure 3.1. Static LFSR Reseeding (using an r-bit LFSR)

The original idea of encoding scan patterns as LFSR seeds was proposed in [Könemann 91]. The encoding efficiency for a set of test cubes is defined as the total number of specified test bits in the test cubes divided by the total number of bits required to encode it [Hellebrand 95a]. So an encoding efficiency of 1 corresponds to the case where the set of test cubes was encoded with the same number of bits as the total number specified bits in all the test cubes. The encoding efficiency for the basic LFSR reseeding approach described in [Könemann 91] is limited by two factors.

1. Linear dependencies in the LFSR - The LFSR must be large enough to yield a solution to the linear equations for all the test cubes in the set. If s_{max} denotes the largest number of specified bits in any test cube in the set, then it has been estimated that the LFSR should have a length of $s_{max}+20$ bits in order to reduce the probability

of not finding a seed for a test cube to less than 10^{-6} [Chen 86], [Könemann 91], due to linear dependencies in the LFSR.

2. Variance in the number of specified bits in test cubes - The number of specified bits in each test cube can vary considerably; however, the size of the LFSR is restricted by the test cube with the largest number of specified bits (s_{max}). So even though most test cubes may have many fewer than s_{max} specified bits, they still are encoded with LFSR seeds having $s_{max}+20$ bits.

So the basic LFSR reseeding approach described in [Könemann 91] requires $s_{max}+20$ bits to encode each test cube regardless of the number of specified bits in the test cube.

Several techniques have been proposed to improve the encoding efficiency including using multiple-polynomial LFSRs [Hellebrand 92], [Venkataraman 93], concatenating test cubes [Hellebrand 95a], using variable-length seeds [Zacharia 95, 96], [Rajski 98a], and using two-dimensional compression combining LFSR reseeding and folding counters [Liang 01].

All the previous approaches for LFSR reseeding have involved *static reseeding*. Static reseeding is defined here as stopping test generation and loading a new seed before resuming test generation. Note that here test generation means the process of expanding a seed into the corresponding full test vector within the scan chains. All the approaches except for test cube concatenation [Hellebrand 95a] stop the test generation after each test vector to load a new seed. The test cube concatenation approach generates a small fixed number of test cubes (e.g., $j=8$) before loading a new seed, but when it does load a new seed, it stops the test generation to do so. All the previous approaches for linear feedback shift register (LFSR) reseeding have also required *full reseeding*. If n is the number of bits that are used for reseeding, then full reseeding is defined here as the case where n equals r for an r-bit LFSR. The variable-length seed method [Rajski 98a] allows for shorter seeds, but the size of the seed is still equal to the size of the variable length LFSR.

The next chapter describes a *partial dynamic* reseeding approach where n is less than r for an r-bit LFSR. Instead of stopping test generation to load the partial seed, the

seed is modified incrementally while the test generation proceeds. Full static forms of LFSR reseeding are shown to be a special case of the new partial dynamic form of LFSR reseeding. In addition to providing better encoding efficiency, partial dynamic LFSR reseeding has a simpler hardware implementation than previous schemes based on multiple-polynomial LFSRs and can generate each test vector in fewer clock cycles.

Chapter 4

Test Vector Encoding Using Partial LFSR Reseeding

This chapter presents a novel test vector compression scheme based on linear feedback shift register (LFSR) reseeding (this work has been published in [Krishna 01]). A new form of LFSR reseeding that provides higher encoding efficiency and hence greater reduction in test data storage requirements is described. As described in the previous chapter, LFSR reseeding is a very practical and powerful approach. Several techniques for improving the encoding efficiency for the basic LFSR reseeding methodology (originally described in [Könemann 91]) have been proposed in [Hellebrand 92, 95a, 95b], [Venkataraman 93], [Zacharia 95, 96], and [Rajski 98a].

Note that all the previous approaches for LFSR reseeding have involved *static reseeding*. Static reseeding is defined here as stopping test generation and loading a new seed before resuming test generation. All the approaches except for test cube concatenation [Hellebrand 95a] stop the test generation after each test vector to load a new seed. The reseeding approach proposed in this chapter is a *dynamic reseeding* method in which the seed is modified incrementally while the test generation proceeds.

All the previous approaches for LFSR reseeding have also required *full reseeding*. If n is the number of bits that are used for reseeding, then full reseeding is defined here as the case where n equals r for an r-bit LFSR. The variable-length seed method [Rajski 98a] allows for shorter seeds, but the size of the seed is still equal to the size of the variable length LFSR. The reseeding approach proposed in this chapter allows for *partial reseeding*, where n is less than r for an r-bit LFSR. Full static forms of LFSR reseeding are shown to be a special case of the new partial dynamic form of LFSR reseeding. In addition to providing better encoding efficiency, partial dynamic LFSR reseeding has a simpler hardware implementation than previous schemes based on multiple-polynomial LFSRs and can generate each test vector in fewer clock cycles.

4.1 Proposed Partial Reseeding Method

The proposed partial reseeding approach is illustrated in Fig. 4.1. Note that an extra XOR gate is included in the feedback of the LFSR. The LFSR length, *r,* is at least s_{max}+20 where s_{max} is the maximum number of specified bits in any test cube. The *r*-bit LFSR is initialized with a starting *r*-bit seed. This initial seed is used to generate the first test cube by running the LFSR for *m* clock cycles (where *m* is the scan length) to fill the scan chains. For the second test cube, the LFSR is run for another *m* clock cycles to generate the next test cube. However, during each of the first *n* clock cycles, a bit is shifted in from the tester and XORed with the feedback of the LFSR. These *n* bits coming in from the tester alter the state of the LFSR and in effect "dynamically reseed" the LFSR. For an *r*-bit LFSR, *n* is significantly smaller than *r*, so it is a "partial reseeding."

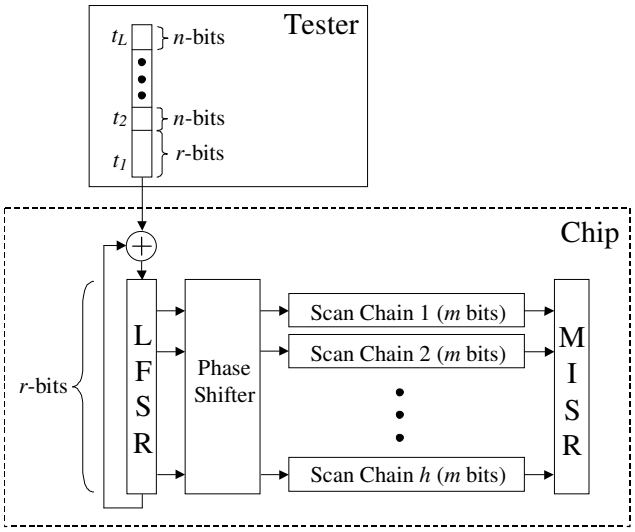


Figure 4.1. Proposed Partial Reseeding Scheme
($n < r$ for an *r*-bit LFSR)

After the first *n* clock cycles, the tester stops shifting in data and the LFSR simply cycles through its normal sequence of states until the scan chains are full. This partial dynamic reseeding process is repeated for each of the subsequent test cubes that are generated by the LFSR. For each test cube, a bit is shifted in from the tester during each of the first *n* clock cycles as the scan chains are filled. The total number of bits required to encode a set of *L* test cubes using the proposed approach with an *r*-bit LFSR is $n \cdot (L-1)+r$. Notice that the number of bits required for encoding is not proportional to

s_{max}. This is a nice property. For all the previously proposed approaches for LFSR reseeding, with the exception of the variable-length seed method [Rajski 98a], the number of bits required for encoding is proportional to s_{max}. The variable-length seed method avoids dependence on s_{max} but at the cost of the extra complexity needed to implement variable size LFSRs.

Note that since the tester loads data onto the chip only for the first n clock cycles of each test cube generation, each test cube receives a fixed number of bits. Moreover, n is less than m, where m is the number of cycles required to generate each test cube within the scan chains. This scheme thus falls under the category of periodically-loaded schemes using a fixed number of free-variables per test cube.

The additional hardware required for the proposed partial reseeding method beyond what is needed for the standard STUMPS architecture [Bardell 82] is just an additional XOR gate in the feedback of the LFSR which is controlled from the tester. There is no need for a multiple-polynomial LFSR or any added complexity. The simplicity of partial reseeding is another nice property that it has.

In dynamic reseeding, the state of the LFSR after the application of test cube t_i carries forward to the generation of the test cube t_{i+1}. This is very beneficial in the following way. Depending on the number of specified bits in test cube t_i, the solution space for the system of linear equations that is solved to find a seed that produces t_i can be very large. Generally, the fewer specified bits in test cube t_i, the larger the solution space is for seeds that generate t_i. With dynamic reseeding, the degrees of freedom in the solution space for t_i can be used to ease the problem of finding a solution for t_{i+1}. By using the degrees of freedom in the solution space for t_i, fewer additional bits need to be shifted in from the tester to find a solution for t_{i+1}. This allows n (the number of bits coming from the tester) to be smaller than r (the size of the LFSR), which results in partial reseeding. In static reseeding methods, the state of the LFSR after applying test cube t_i is completely overwritten when a new seed is loaded for test cube t_{i+1}. Hence, the degrees of freedom in the solution space for test cube t_i are completely wasted. With dynamic reseeding, the degrees of freedom in the solution space for test cube t_i are preserved and can be used for

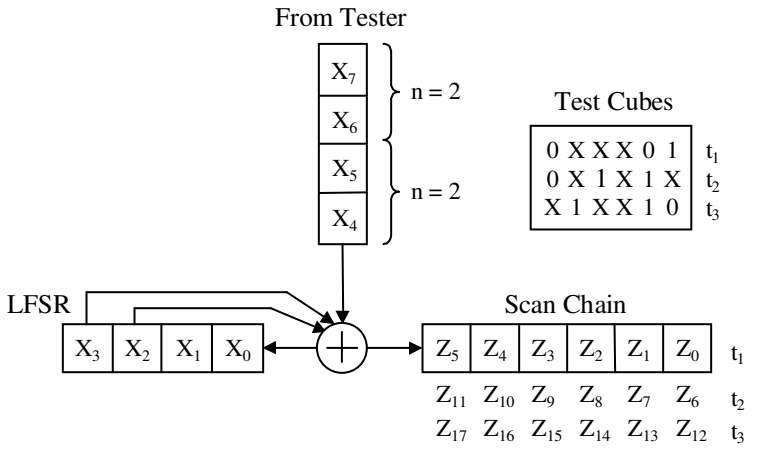
solving the linear equations for subsequent test cubes. This results in a much better encoding efficiency.

For partial dynamic reseeding, to maximally exploit the ability to use the degrees of freedom in the solution space of the previous test cube when solving the linear equations for next test cube, the test cubes should be ordered in the following way. The test cubes with the most number of specified bits should be interleaved with the test cubes with the fewest number of specified bits (e.g., have the least specified test cube followed by most specified test cube, followed by second least specified test cube, followed by second most specified test cube, etc.). This eases the burden on solving the linear equations by matching the larger solution spaces for a preceding test cube with the most specified (i.e., hardest to solve test cubes), and the smaller solution spaces for a preceding test cube with the least specified (i.e., easiest to solve) test cubes. By so doing, the value of n can be minimized.

So far in this chapter, dynamic reseeding has been described with the data being shifted in from the tester at the same time as data is being shifted from the LFSR into the scan chains *("dynamic reseeding concurrent with scan chain loading")*. However, for some applications, it may be desirable to load the scan chains at a faster clock rate than the tester clock rate. In that case, dynamic reseeding can also be implemented by shifting in the n bits of data from the tester and cycling the LFSR without loading the scan chains *("dynamic reseeding before scan chain loading")*. After all n bits have come in from the tester and dynamically reseeded the LFSR, then the scan chains can be loaded from the LFSR at a clock rate that is faster than the tester clock rate since no more interaction with the tester is required for that test vector. Dynamic reseeding before scan chain loading retains all the same properties as dynamic reseeding concurrent with scan chain loading (the only difference is the phase of the linear equations). Note that dynamic reseeding before scan chain loading with n equal to r is equivalent to conventional full static reseeding. If n equals r, then the state of the r-bit LFSR can be completely controlled by the $n=r$ bits coming in from the tester. The LFSR can be forced into any state by the proper selection of the $n=r$ bits coming from the tester. Therefore, full static reseeding can be considered a special case of partial dynamic reseeding.

4.2 Forming and Solving Linear Equations for Partial Reseeding

Now that partial reseeding has been described, the next issues are how to form and solve the linear equations for the $n \cdot (L-1)+r$ bits that are stored on the tester in order to generate a set of L test cubes, and how to choose the minimum value of n that will result in a solution. Forming the linear equations is done by representing the $n \cdot (L-1)+r$ bits stored on the tester with symbols and symbolically simulating the LFSR operation to generate the linear equations for each specified bit in the test cubes. A small example is shown in Fig. 4.2. A 4-bit LFSR is used to generate three test cubes with $n = 2$. In this case, $n \cdot (L-1)+r = 2(3-1)+4 = 8$. So the test cubes are encoded with 8 bits of data that are symbolically represented by X_0 through X_7. The scan chain is 6 bits long, so there is a total of 18 bits in the 3 test cubes. The equations for these 18 bits are represented by Z_0 through Z_{17}. Note that for simplicity, the example shows an LFSR feeding a single scan chain; however without loss of generality, the same procedure would apply for an LFSR feeding multiple scan chains.



Test Cube t_1	Test Cube t_2	Test Cube t_3
$Z_0 = X_2 + X_3$	$Z_6 = X_1 + X_2 + X_3 + X_4$	$Z_{12} = X_2 + X_4 + X_6$
$Z_1 = X_1 + X_2$	$Z_7 = X_0 + X_1 + X_2 + X_5$	$Z_{13} = X_1 + X_5 + X_7$
$Z_2 = X_0 + X_1$	$Z_8 = X_0 + X_1 + X_2 + X_3$	$Z_{14} = X_0 + X_4$
$Z_3 = X_0 + X_2 + X_3$	$Z_9 = X_0 + X_1 + X_3 + X_4$	$Z_{15} = X_2 + X_3 + X_4 + X_5 + X_6$
$Z_4 = X_1 + X_3$	$Z_{10} = X_0 + X_3 + X_4 + X_5$	$Z_{16} = X_1 + X_2 + X_4 + X_5 + X_6 + X_7$
$Z_5 = X_0 + X_2$	$Z_{11} = X_3 + X_5$	$Z_{17} = X_0 + X_1 + X_4 + X_5 + X_7$

Figure 4.2. Example of Forming Equations for Partial Reseeding

Once the linear equations have been formed, they can be efficiently solved using Gauss-Jordan elimination. For the example in Fig. 4.2 where test cubes t_1, t_2, and t_3, are *0XXX01*, *0X1X1X*, and *X1XX10*, respectively, a solution can be obtained by solving the equations for the bits with specified values. One solution would be $X_0=1$, $X_1=1$, $X_2=1$, $X_3=0$, $X_4=1$, $X_5=0$, $X_6=0$, $X_7=0$. Note that the "+" operation shown in Fig. 4.2 represents modulo-2 addition.

The larger the value of n, the more likely there is to be a solution to the linear equations. If the value of n is too small, a solution may not exist. If the value of n is greater than or equal to $s_{max}+20$, then there is an extremely high probability of finding a solution. The minimum value of n for which a solution might reasonably be found would be s_{avg} which is the average number of specified bits per test cube. One strategy for quickly finding a small value of n that gives a solution is to do a binary search between s_{avg} and $s_{max}+20$. This would require $\lceil \log_2(s_{max}-s_{avg}+20) \rceil$ iterations. Each iteration involves forming and trying to solve the system of linear equations.

One disadvantage of partial dynamic reseeding compared with full static reseeding is that the computation time for solving the linear equations is longer. In full static reseeding, the linear equations for each test cube can be solved independently. In partial dynamic reseeding, the linear equations for the test cubes need to be solved all together. Although this results in a more efficient solution, it may not scale well for large test sets. However, the solution for this problem is very simple. For large test sets, the test cubes can be partitioned into smaller subsets of k test cubes each. Partial dynamic reseeding can then be done for each subset of k test cubes. After each subset of k test cubes is generated by the LFSR with partial dynamic reseeding, the seed of the LFSR is re-initialized before the next subset of k test cubes are generated by the LFSR with partial dynamic reseeding. Thus the linear equations for each subset of k test cubes can be formed and solved independently. The value of k can be chosen based on the amount of computation time that is acceptable. This provides a very easy tradeoff between computation time and the optimality of the result. Note that in the degenerate case where k is equal to 1, partial dynamic reseeding reduces to full static reseeding. It should be

noted that generating and solving the system of linear equations for partial dynamic reseeding can be done in polynomial time (it is not exponential), and the procedures are very efficient and fast. Our experiments indicate that values of k in the order of hundreds can be processed in a few hours. So in many cases, if the number of test cubes is in the order of hundreds, it may not be necessary to partition the problem.

Partitioning in some cases can actually be used to obtain a better solution if there is a large variance in the number of specified bits in the test cubes. One strategy for partitioning would be to group the test cubes with the largest number of specified bits in one partition, and the test cubes with a smallest number of specified bits in another partition. The partition with the larger number of specified bits would end up with a larger value of n, and the partition with the smaller number of specified bits would have a smaller value of n. The encoding efficiency for the two separate partitions may be higher than the encoding efficiency of processing all of the test cubes together in one partition. Note that the tester program required for handling multiple values of n would be more complex than for a single value of n. Experiments done with the ISCAS 89 benchmark circuits showed that in some cases the total test storage requirements were slightly better with partitioning, and in some cases they are slightly worse. But overall, partitioning did not make too much difference in the results.

4.3 Experimental Results

Experiments were performed on the largest ISCAS 89 benchmark circuits [Brglez 89]. For each circuit, 10,000 pseudo-random patterns were applied using the LFSR to detect the easy faults. ATPG was performed to generate test cubes for the remaining faults. Partial reseeding was then used to encode the set of test cubes. The results are shown in Table 4.1. The last column shows the compression ratio that is achieved with partial reseeding which is the ratio of the test storage requirements for the unencoded test vectors (i.e., simply storing the test vectors themselves on the tester) compared with the test storage requirements using partial reseeding. As can seen, partial reseeding generally provides an order of magnitude reduction in test storage requirements. Table 4.2 shows a

comparison of partial reseeding with previous reseeding schemes (all of which are based on full static reseeding). Each of the reseeding schemes was used to encode the set of test cubes in Table 4.1. The exact same set of test cubes is encoded in each case to provide an "apples to apples" comparison. As can be seen, partial reseeding clearly provides the highest encoding efficiency.

Table 4.1. Results for Partial Reseeding After Pseudo-Random Sequence of 10,000 Patterns

Circuit		Num Test Cubes	Num Specified Bits	s_{max}	LFSR size	Bits Per Vector *(n)*	Test Data		
Name	Scan Elements						Test Storage	Encoding Efficiency	Compression Ratio
s5378	214	30	493	18	38	16	502	.982	12.8
s9234	247	138	4674	61	81	36	5013	.932	6.8
s13207	700	157	2824	24	44	19	3008	.938	36.5
s15850	611	167	5092	38	58	31	5204	.978	19.6
s38417	1664	340	23984	85	105	72	24513	.978	23.1
s38584	1464	62	2848	55	75	47	2942	.968	30.9

Table 4.2. Comparison of Reseeding Schemes for Same Test Set

Circuit Name	Standard LFSR Reseeding [Könemann 91]			Test Cube Concatenation [Hellebrand 95a]			Variable-Length Seeds [Rajski 98a]			Proposed Partial Reseeding		
	LFSR size	Total Bits	Eff.	LFSR size	Total Bits	Eff.	LFSR size	Total Bits	Eff.	LFSR size	Total Bits	Eff.
s5378	38	1140	.432	18	570	.865	18	658	.749	38	502	.982
s9234	81	11178	.418	61	5332	.877	61	5364	.871	81	5013	.932
s13207	44	6908	.409	24	3925	.719	24	3609	.782	44	3008	.938
s15850	58	9686	.526	38	6513	.782	38	5927	.859	58	5204	.978
s38417	105	35700	.672	85	29240	.820	85	25684	.934	105	24513	.978
s38584	75	4650	.612	55	3472	.820	55	3158	.902	75	2942	.968

Chapter 5

Hybrid BIST Using an Incrementally Guided LFSR

The test data volume required for testing increasingly complex system-on-chip (SOC) designs continues to grow rapidly and is outstripping the capabilities of ATE (automated test equipment). While both the tester storage and the test application time are significant contributors towards the cost of testing a chip, sometimes it may be necessary to reduce the tester storage further even at the expense of incurring additional test time. Based on the test resources that are available, it may turn out to be cheaper to allow the chip some more time on the tester rather than go to the expense of obtaining additional tester storage or even upgrading to a newer tester. In such a scenario, a scheme having the ability to tradeoff tester storage versus test time can be very useful. This chapter describes a periodically-loaded scheme (this work has been published in [Krishna 03a]) that provides the ability to tradeoff tester storage versus test time based on the test resources that are available.

The previous chapter described a periodically-loaded scheme with a fixed amount of data per test cube (test vector where the unspecified bits are left as don't cares) being stored on the tester. If the longest scan chain is m bits long, then after exactly m clock cycles, the required test cube can be embedded within the test vector generated in the scan chains. The linear equations are generated and solved in a manner such that it is possible to embed a test cube within the test vector generated in the scan chains. Thus if there are T test cubes then these test cubes will be embedded within the first T vectors that are shifted into the scan chains. This scheme is able to significantly reduce the tester storage as has been shown in the experimental results.

The scheme proposed in this chapter provides the ability to tradeoff tester storage versus test time by embedding the deterministic test cubes within a pseudo-random BIST sequence. Instead of embedding the T test cubes within the first T test vectors, the test cubes are embedded within the first L ($L \geq T$) pseudo-random test vectors generated by the linear expansion network. Note that L is also called the test length. The data stored

on the tester is used to incrementally guide the LFSR such that the T test cubes can be embedded within the given test length (L). The bigger the value of L, the smaller the data that needs to be stored on the tester. A longer test length also translates to a longer test application time since more test vectors have to be generated within the scan chains. By choosing an appropriate test length, the data to be stored on the tester can be made to fit within the available tester memory. Thus the proposed scheme allows the user to tradeoff the tester storage versus the test application time by manipulating the test length within which to embed the test cubes. Note that for the case where L equals T, the proposed scheme is similar to the scheme described in the previous chapter. Thus the proposed scheme can be thought of as a more generalized version of the partial reseeding scheme.

The features of the proposed scheme also allow it to be classified as a "hybrid BIST" scheme. A "hybrid BIST" scheme is one in which some external data from the tester is combined with the BIST hardware to achieve the desired fault coverage. A hybrid BIST scheme reduces the test data stored on the tester compared with full external testing, but it does not require as much hardware overhead as full stand-alone BIST. In terms of test application time, a hybrid BIST scheme requires the application of more test vectors than conventional external testing, but less than that required for pseudo-random BIST.

A simple hybrid BIST scheme is to use a STUMPS architecture [Bardell 82] to apply pseudo-random patterns to detect the random-pattern-testable faults, and then use deterministic vectors from the tester to detect the random-pattern-resistant (r.p.r.) faults. Results for using this scheme on large industrial designs have shown that it only achieves around a 30-50% reduction in tester storage requirements compared with conventional external testing [Hetherington 99], [Pressly 99]. More sophisticated hybrid BIST schemes have been developed including using hybrid patterns [Das 00], folding counters [Hellebrand 00], two-dimensional compression [Liang 01], weighted pattern testing [Jas 01], and RESPIN [Dorsch 01a, 01b].

The scheme proposed in this chapter is a new hybrid BIST scheme that uses an "incrementally guided LFSR." The hardware overhead is very small. A conventional STUMPS architecture is used with only a small modification to the feedback of the LFSR which allows the tester to incrementally guide the LFSR so that it can embed test cubes

that detect the r.p.r. faults in the pseudo-random sequence. Compared with external testing, the proposed approach achieves dramatic reductions in tester storage requirements while using very simple on-chip hardware. Results indicate that the proposed approach provides very attractive tradeoffs between test length and tester storage requirements.

5.1 Overview of Proposed Method

The proposed method embeds deterministic test cubes (test vectors where the unspecified bits are left as don't cares) that detect the r.p.r. faults in the pseudo-random sequence generated by an LFSR. The architecture for the proposed method is shown in Figure 5.1. A conventional STUMPS [Bardell 82] architecture is used except that an extra exclusive-or (XOR) gate is added in the feedback of the LFSR. The extra exclusive-or gate is controlled by the tester and allows the tester to incrementally guide the LFSR. Some control logic is required to obtain data from the tester only during the first few clock cycles of every test vector generation. In order to be able to generate the test cubes with a high degree of probability, the LFSR length, r, is at least s_{max}+ 20 where s_{max} is the maximum number of specified bits in any test cube. The r-bit LFSR can be initialized with a starting r-bit state, or it can start from the reset state. Since the length of the scan chains is m-bits, the LFSR needs m clock cycles to generate a test vector within the scan chains. During the first clock cycle while a test vector is being shifted into the scan chains from the LFSR, one bit of deterministic data is obtained from the tester and XORed with the feedback of the LFSR. During the remaining m-1 clock cycles while the LFSR is filling the scan chains with the test vector, the tester does not shift any data into the LFSR. This process is repeated for all the test vectors that are generated by the LFSR. So the test data storage requirement is one bit per test vector applied to the circuit-under-test (CUT). While the LFSR shifts in the test vector, the output response contained in the scan chains is shifted out into a multiple-input signature register (MISR).

The one deterministic bit coming in from the tester for each test vector can be chosen to be either a zero or a one. It is essentially a free-variable whose value can be chosen in

a way that incrementally guides the LFSR towards a state in which it will produce a required test cube. The impact of each free-variable on the subsequent test vectors that are applied to the CUT can be determined by symbolic simulation of the LFSR.

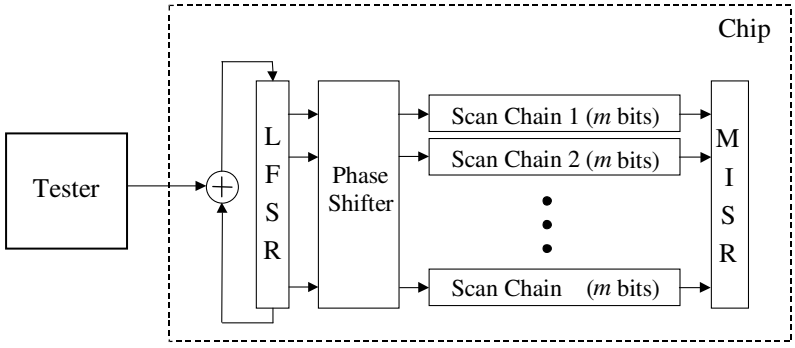


Figure 5.1. Architecture for Pseudo-Random BIST with Incrementally Guided LFSR

Consider the example in Figs. 5.2-5.4. Assume that the LFSR starts in state *1011*, and one bit comes in from the tester which is represented by the free-variable *a*. The LFSR then runs in autonomous mode to fill the scan chains with the first test vector. The first test vector is computed by symbolic simulation and is shown in Fig. 5.3. In the first clock cycle, the bits (1,0,1) get shifted into the first bit-slice of the scan chains. During the next clock cycle, the bits (a,1,0) get shifted into the scan chains. At the end of 4 clock cycles, the scan chains are completely filled with the first test vector which is in terms of the free-variable *a*. Then the next bit comes in from the tester which can be represented by the free-variable *b*. The LFSR is then symbolically simulated to determine the second test vector (which now is in terms of both free-variables *a* and *b*). This process continues as more and more free-variables are introduced. At some point, it may be possible to embed a test cube for an r.p.r. fault by assigning values to some of the free-variables. For example, the test cube X0XX, X1X0, 1XX0 could be embedded in the third test vector by assigning the values *a=0* and *b=0*. Note that the equations for each bit in the test vector are linear (sum modulo-2), so finding an assignment to the free-variables to embed the test cube can be solved very efficiently using a linear equation solver [Könemann 91]. By using the computed values for the free-variables, the data from the tester incrementally guides the LFSR into a state in which the test cube for the r.p.r. faults gets

applied to the CUT. This process of adding more free-variables and embedding additional test cubes for hard faults can be repeated until sufficiently high fault coverage is achieved. This is the basic approach for BIST based on an incrementally guided LFSR.

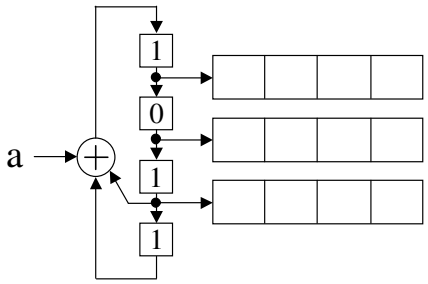


Figure 5.2. LFSR Configuration for Example

Vector 1			
1	1	a	1
1	a	1	0
a	1	0	1

Vector 2			
a+b	0	a’	a’+b
0	a’	a’+b	1
a’	a’+b	1	1

Vector 3			
a+c	a+b	a’	b+c
a+b	a’	b+c	a+b
a’	b+c	a+b	0

Figure 5.3. Symbolic Simulation of LFSR in Fig. 5.2

Vector 1			
1	1	0	1
1	0	1	0
0	1	0	1

Vector 2			
0	0	1	1
0	1	1	1
1	1	1	1

Vector 3			
c	0	1	c
0	1	c	0
1	c	0	0

Figure 5.4. Vectors after Assignment of a=0 and b=0

The approach is very efficient because there are a lot of degrees of freedom for embedding the test cubes for the r.p.r. faults. The degrees of freedom are the following: selecting the values of the free-variables, selecting which test cube to embed, and selecting which test vector in which to embed the test cube.

So far we have been assuming that only one bit of data will come in from the tester for each test vector. However, this may result in too long of a test length for this BIST

scheme. The test length for this BIST scheme can be reduced by increasing the number of channels by which data is brought in from the tester. For example, if we used 3 channels, we could bring in 3 bits of data from the tester during the first clock cycle each time a test vector is shifted into the scan chains. This would increase the rate at which free-variables are introduced and hence reduce the overall BIST test length because the test cubes for the r.p.r. faults could be embedded sooner. The degree of freedom in choosing which test vector in which to embed each test cube would be reduced since the test length is reduced, and hence the encoding efficiency would be slightly degraded. However, this provides a very easy way for trading off test data bandwidth requirements, test data storage requirements, and test length.

The proposed scheme brings in data from the tester only during the first clock cycle each time a test vector is shifted into the scan chains. Thus it is a periodically-loaded scheme. Even though each test vector receives the same number of bits from the tester, different test cubes might require a different number of test vectors to go by before they can be embedded within a suitable test vector. Thus this scheme is a periodically-loaded scheme using a variable number of bits per test cube.

Note that the proposed method can also be used to implement traditional "stand-alone" BIST where no data comes from the tester. In this case, a ROM would be used. The ROM would supply the deterministic data to guide the LFSR instead of having that data coming from an external tester.

5.2 Determining Data on Tester

Given the test cubes for the r.p.r. faults and the maximum allowable test length (L), this section describes how to obtain the data that is stored on the tester. Since the test cubes will be embedded within a pseudo-random sequence of test vectors, only the test cubes for the r.p.r. faults are required.

An important parameter that needs to be determined for the proposed method is the number of bits of data that will come in from the tester for each test vector. If n bits of data need to be brought from the tester for each test vector, then n channels are required between the tester and the LFSR in order to XOR these n bits of data with the contents of

the LFSR during the first clock cycle. These n bits can be XORed with the contents of the LFSR at regularly spaced tap points within the LFSR.

The amount of storage required on the tester for storing the test data can be obtained from the test length and the number of channels from the tester. For a test length of L vectors with n bits coming from the tester per vector, the number of bits that need to be stored on the tester is $L*n$. This provides an upper bound on the amount of tester storage. If the test cubes can be embedded in less than L vectors, then the amount of tester storage is reduced.

Given the set of test cubes T and the test length L, an initial estimate can be found for the number of bits (n) to be used per test vector. Given the total number of specified bits within the set of test cubes, the initial estimate of n is given by:

$$n = \lceil \textit{Total number of specified bits / Test Length} \rceil$$

If this turns out to be 0, then n is assigned the value 1.

Based on this initial estimate of n, the proposed method tries to embed the test cubes for the r.p.r. faults within the maximum allowable test length L. If the test cubes cannot be embedded within the test length L, then more data needs to be introduced per vector in order to embed the given test cubes. Thus the value of n has to be increased, and this process is repeated with the higher value of n. If the test cubes can be embedded within L test vectors, then fault simulation is done in order to ensure that this sequence of vectors with embedded test cubes provides sufficient fault coverage. If the fault coverage is not satisfactory, then ATPG can be performed on the undetected faults to obtain a set of *top-up* cubes. The T test cubes and the *top-up* cubes are then together embedded within the given test length L. Once this is done, the amount of test data that needs to be stored on the tester is given by $L*n$.

The steps of this procedure are as follows:

1. Fault simulate L pseudo-random vectors
2. Do ATPG for faults not detected by the pseudo-random vectors to obtain set of test cubes, T
3. Embed test cubes T with n channels to obtain embedded vectors, L'
4. If $|L'| > |L|$, then increment n and go to step 3
5. Fault simulate L' embedded vectors
6. If the fault coverage is satisfactory, then the procedure is done.
7. Do ATPG for faults not detected by L' and add the resulting test cubes to the set T
8. Go to step 3

The procedure iterates until the fault coverage is satisfactory. Generally this requires only one iteration.

5.3 Improving Compression Using Lookahead

The compression achieved using the basic hybrid BIST method described earlier can be improved further by performing a *lookahead* while generating the test data. The idea is that the data from the tester is used to guide the LFSR towards a test cube in a manner that also facilitates the embedding of subsequent test cubes. Thus once a test cube has been embedded, the subsequent test cube can be embedded within fewer cycles than the previous approach. The previous approach is equivalent to using a *lookahead* of 0, since none of the subsequent test cubes are considered while embedding the current test cube. A *lookahead* of k means that the equations for the subsequent k test cubes are considered while solving the equations for the current test cube.

With this feature, the degrees of freedom in the solution space of the linear equations for a test cube t_i are used to help solve for a subsequent test cube t_{i+1} since the free-variables occurring in the equations of t_i are used to ease the problem of finding a solution for t_{i+1}. Since the subsequent test cube is being embedded using fewer cycles, fewer bits have to be stored on the tester to guide the LFSR towards t_{i+1}. This helps to improve the compression achieved using this BIST scheme.

This procedure for a *lookahead* of k is described as follows. As explained in Fig. 5.4, each test vector is determined by a symbolic simulation of the LFSR using the free-variables from the tester. After it is determined that a test cube t_i can be embedded by solving the linear equations, an assignment of values to the free-variables is not done immediately. Instead, more free-variables are introduced until another k test cubes can be embedded within the vectors generated by the LFSR. This is done by concatenating the equations for the k test cubes with the equations for t_i to form a system of linear equations. A solution to this system of equations indicates that it is possible to embed the subsequent k test cubes using all the free-variables introduced until then. Based on this solution, an assignment of values is done to *only the free-variables that occur in the equations for* t_i. The values assigned to these free-variables are then used to update the equations for the subsequent k test cubes as well as the state of the LFSR. Thus the assignment of values to the free-variables occurring in t_i is done in a manner that helps to solve for the next k test cubes.

The concept behind this can be shown with a simple example. Let a and b be two free-variables occurring in the equations for test cube t_i, and {01,10} be two possible assignments to these variables. For a *lookahead* of 0, either of these two assignments can be used. But if a *lookahead* greater than 0 is used, it is possible that assigning 01 to the variables might help to embed test cube t_{i+1} earlier than would have been possible by using the assignment 10. This degree of freedom is utilized by concatenating and solving together the equations of t_i and t_{i+1}. This ensures that the variables in t_i are assigned values which also help to solve the equations of t_{i+1} and thus guide the LFSR towards generating t_{i+1} earlier than would have been possible with a *lookahead* of 0.

5.4 Experimental Results

Experiments were performed on the largest ISCAS 89 benchmark circuits. Given different values of n (the number of bits that are used per vector), the proposed hybrid BIST scheme was used to embed test cubes in a pseudo-random sequence to detect all non-redundant faults. The resulting test length and tester storage requirements are

graphed in Fig. 5.5 in a log-log plot. A *lookahead* of 0 was used in generating these results. As can be seen, the proposed method provides a tradeoff between test length and tester storage requirements. As the test length increases, the tester storage requirements are reduced. As n is increased, the test length reduces towards that of conventional external deterministic testing. As n is decreased, the tester storage requirements drop and if the test length were made long enough to detect all faults, the tester storage requirements would go to zero as in conventional stand-alone BIST. A designer can determine what test length is desired and use the proposed method to minimize the tester storage requirements for that test length. This is achieved with very little hardware overhead beyond what is needed for the STUMPS architecture.

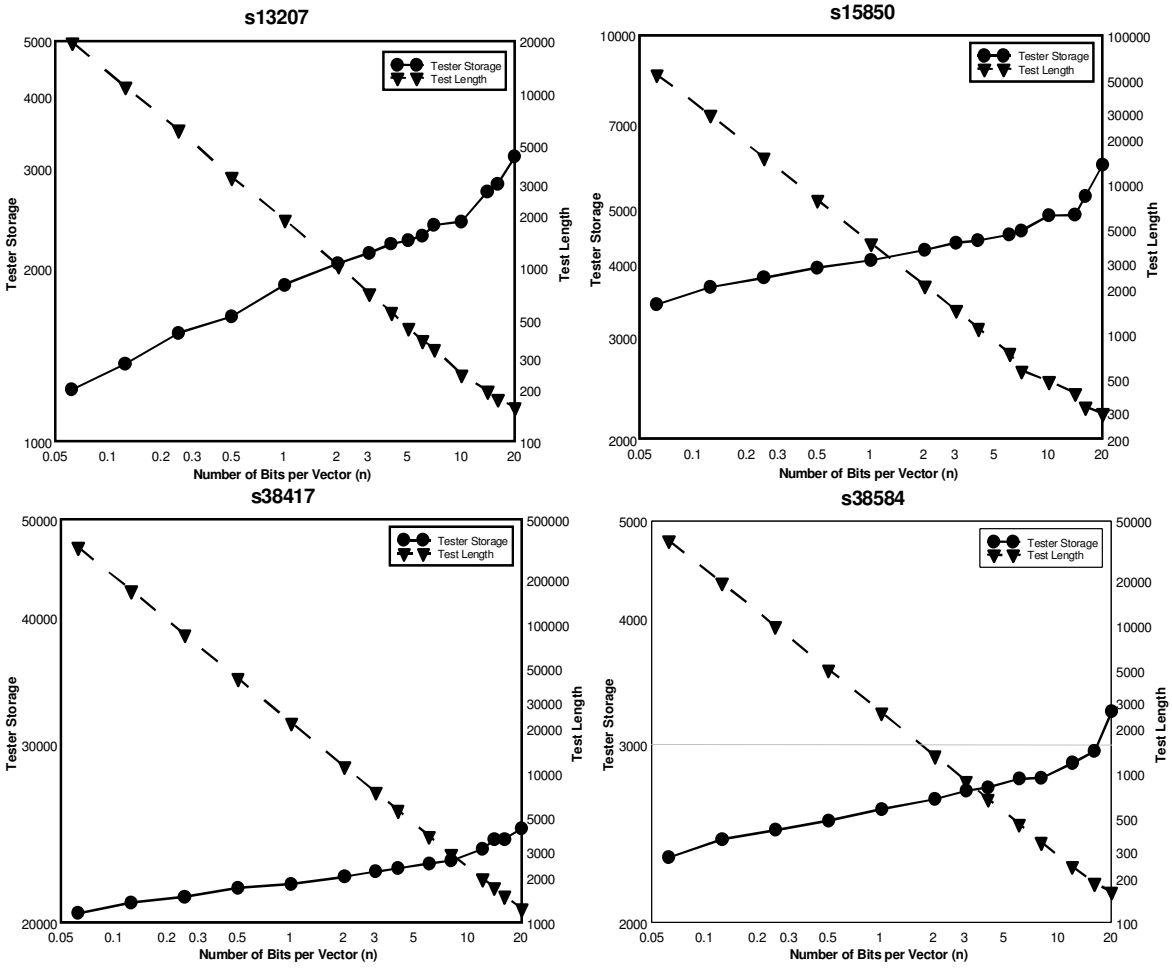


Figure 5.5. Graphs of Tester Storage and Test Length Versus Number of Bits from Tester Used per Vector for ISCAS Benchmark Circuits

Table 5.1 shows results for the proposed scheme for different values of *lookahead*. The number of scan elements is shown for each circuit (it is assumed that their primary inputs are controlled by a scan chain). The next column shows the different number of bits per vector (n) that are used for each circuit. If n is less than 1, it indicates that a single bit is received from the tester every ($1/n$) test vectors. Thus the value n=0.25 means that at the beginning of every fourth test vector, a single bit from the tester is XORed with the contents of the LFSR. For each circuit, results are shown for three different values of n using a *lookahead* of 0, 1, and 4. For each combination of n and the *lookahead* factor, the test length (L) required to embed the top-up cubes is shown, followed by the amount of data that needs to be stored on the tester. As can be seen, using a *lookahead* of 1 and 4 did reduce the storage requirements for some of the circuits, though not significantly.

Table 5.1. Results for Proposed Scheme using different values for lookahead

Circuit		Bits per Vector (n)	*lookahead* = 0		*lookahead* = 1		*lookahead* = 4	
Name	Scan Elements		Test Len (L)	Tester Storage	Test Len (L)	Tester Storage	Test Len (L)	Tester Storage
s13207	700	0.25	6,104	1,526	6,078	1,519	6,078	1,519
		1	1,856	1,856	1,818	1,818	1,818	1,818
		4	553	2,212	545	2,180	537	2,148
s15850	611	0.25	15,216	3,804	14,985	3,746	14,985	3,746
		1	4,124	4,124	4,115	4,115	4,115	4,115
		4	1,103	4,412	1,096	4,384	1,096	4,384
s38417	1664	0.25	85,093	21,273	85,093	21,273	85,093	21,273
		1	21,855	21,855	21,846	21,846	21,846	21,846
		4	5,623	22,492	5,598	22,392	5,582	22,328
s38584	1464	0.25	9,906	2,476	9,816	2,454	9,816	2,454
		1	2,592	2,592	2,562	2,562	2,562	2,562
		4	685	2,740	676	2,704	674	2,696

Table 5.2 shows a comparison of the proposed scheme with using normal pseudo-random BIST to apply 10,000 pseudo-random patterns followed by top-up vectors (to detect the remaining undetected faults) and a comparison with using RESPIN [Dorsch 01a, 01b]. The storage requirements for each scheme are shown. The results for RESPIN were taken from [Dorsch 01a] and [Dorsch 01b]. In [Dorsch 01a], no special

ATPG is used; however, in [Dorsch 01b], a special ATPG tailored for RESPIN is used. Note that the test length and test storage for RESPIN are the same. The results for the proposed scheme are shown for a *lookahead* of 0. For each value of *n*, the test length (L) required to embed the top-up cubes is shown, followed by the amount of data that needs to be stored on the tester. Lastly, the percentage compression that is achieved using the proposed scheme compared with pseudo-random BIST followed by top-up vectors is shown. As can be seen, the storage required for the proposed scheme is less than the storage required for the basic RESPIN architecture in [Dorsch 01a] for all the cases. When compared with the RESPIN plus tailored ATPG technique in [Dorsch 01b], the proposed scheme shows better results for some of the circuits. One of the advantages of RESPIN is that it facilitates the use of a tailored ATPG; however, the runtime for that ATPG can be longer than that of conventional ATPG. The results for the proposed scheme were for using conventional ATPG.

Table 5.2. Comparison of Results for Proposed Scheme with other Methods

Circuit		BIST followed by Top-up Vectors		RESPIN [Dorsch 01a]	RESPIN + ATPG [Dorsch 01b]	Proposed Hybrid BIST Scheme (*lookahead* = 0)			
Name	Scan Elems.	Num Vectors	Tester Storage			Bits per Vector (*n*)	Test Len (*L*)	Tester Storage	% Compress
s13207	700	157	109,900	1,963	1,672	0.25 1 4	6,104 1,856 553	1,526 1,856 2,212	98.6 98.3 97.9
s15850	611	167	102,037	5,244	2,872	0.25 1 4	15,216 4,124 1,103	3,804 4,124 4,412	96.3 95.9 95.7
s38417	1664	340	565,760	31,656	8,412	0.25 1 4	85,093 21,855 5,623	21,273 21,855 22,492	96.2 96.1 96
s38584	1464	62	90,768	3,466	2,927	0.25 1 4	9,906 2,592 685	2,476 2,592 2,740	97.3 97.1 96.9

Chapter 6

Reducing Test Data Volume Using LFSR Reseeding with Seed Compression

In this chapter a new lossless test vector compression scheme is presented which combines LFSR reseeding and statistical coding in a powerful way (this work has been published in [Krishna 02]). As described in the earlier chapters, test vectors can be encoded as LFSR seeds by solving a system of linear equations based on the feedback polynomial of the LFSR. Since the seeds are much smaller than the full test vectors, the amount of test data can be reduced by an order of magnitude or more. However, the linear dependencies in the LFSR and the variance in the number of specified bits in each test cube limit the encoding efficiency of LFSR reseeding. In this chapter, a new approach is proposed for improving the encoding efficiency of LFSR reseeding that involves compressing the seeds using a statistical code. Depending on the size of the LFSR and the number of specified bits in the test vector, the solution space of the linear equations can be quite large. The scheme described in this chapter takes advantage of this large solution space to find LFSR seeds that can be efficiently encoded using a statistical code. The solution space for the seeds has the nice property of being a linear space, and the proposed approach uses some heuristics to traverse the linear solution space to optimally choose the seeds that can be encoded using the compression code. The combination of test vector compression using LFSR reseeding and then further compression of the seeds using statistical encoding provides very high encoding efficiency.

The proposed scheme can be used either for applying a fully deterministic test set or for mixed-mode BIST, and it can be used in conjunction with other variations of LFSR reseeding that have been previously proposed (including the one described in chapter 4) to further improve the encoding efficiency. It is very effective even for the basic LFSR reseeding scheme where each test cube is encoded with s_{max}+20 bits because the large solution space for each seed enables it to be highly compressed using the proposed

procedure. For sake of simplicity, the proposed seed compression methodology will be described here for the basic LFSR reseeding scheme, but note that it can also be used with any of the other more efficient LFSR reseeding schemes to provide even better results.

Statistical encoding can be applied directly on the test cubes themselves as was described in [Jas 99]. However, using statistical encoding on LFSR seeds instead of the test cubes provides two advantages. The first is that the number of blocks of data that need to be encoded is dramatically reduced because the LFSR seeds are much smaller than the test cubes (typically an order of magnitude smaller). The second is that the specified bits in the test cubes cannot be easily changed (it requires re-running the ATPG with special constraints which is time consuming), whereas the specified bits in the LFSR seeds can be easily altered by simply choosing a different seed in the linear solution space. The degrees of freedom in choosing the LFSR seed allows a much more efficient encoding compared with encoding the test cubes directly. This is very apparent in the experimental results shown in Section 6.4.

6.1 Architectures for LFSR Reseeding with Seed Compression

Two architectures for implementing the proposed test vector compression scheme are presented here. One has the LFSR integrated as part of the scan chains while the other uses a separate small LFSR. Each architecture has certain advantages and is better suited for particular applications.

6.1.1 Integrated LFSR

LFSR reseeding involves loading seeds one at a time into an LFSR and running the LFSR in autonomous mode to fill the scan chains. For encoding a fully deterministic test set that covers all the faults (i.e., no pseudo-random patterns are used), each test cube is generated by decompressing an LFSR seed. Since there is no need to save the state of the LFSR after it has decompressed a seed to fill the scan chains, its contents can be overwritten by the output response during the capture cycle. So in this case, the LFSR

can be implemented by configuring part of the scan chains to perform the LFSR functionality, as illustrated in Fig. 6.1. By using the scan cells themselves to form the LFSR, there is no need for a separate LFSR and hence the hardware requirements are reduced.

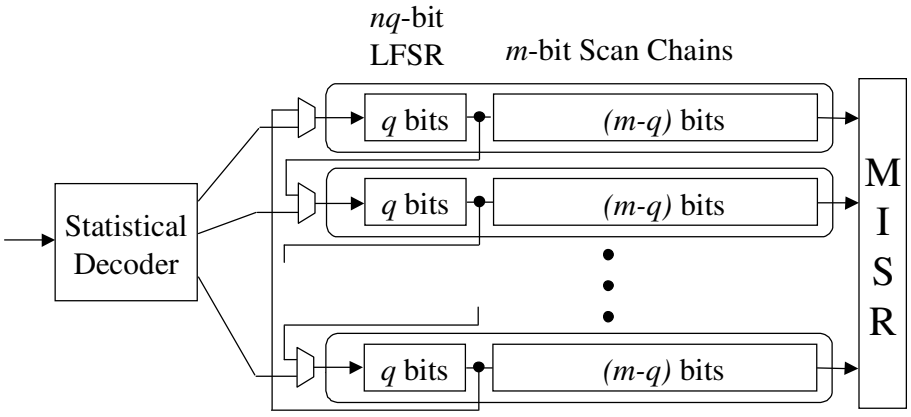


Figure 6.1. Architecture with Integrated LFSR

The same number of scan cells, q, from each scan chain is used to form the LFSR. If there are n scan chains, then the total length of the LFSR is nq. The value of q is selected so that nq is as close to s_{max}+20 as possible (equal to or slightly greater) where s_{max} is the maximum number of specified bits in any test cube. This is done so that it is possible to solve the linear equations for all the test cubes with very high probability [Könemann 91].

The scan architecture has 3 modes: scan mode, system mode, and decompression mode. The scan mode and system mode are the same as for all scan chains. In the decompression mode, the first q scan cells of each m-bit long scan chain are configured together as one long LFSR while the remaining m-q scan cells in each scan chain act as they normally do in scan mode (they are essentially fed from tap points in the LFSR). Note that there is no need for a phase shifter as the tap points are generally spaced apart sufficiently to eliminate correlations (assuming a modular LFSR structure [Bardell 87]). The LFSR is run in autonomous mode for m-q clock cycles to fill up the last m-q scan cells in each scan chain. The linear equations used to solve for the seed are formed so that the final contents of all the scan cells (including those that are part of the LFSR) will

contain the test cube that needs to be applied to the circuit-under-test (CUT) at that point. The scan chain can then be put into system mode to apply the capture cycle so that the output response of the CUT is loaded into the scan chains.

The process of loading a seed into the LFSR is done as follows. The tester sends data to the statistical decoder which decodes it and shifts it into the first q bits of the scan chains (while the scan architecture is in scan mode). The decoding process will be described in more detail in Sec. 6.2. As the decoder shifts the first q bits into the scan chains, q bits of the output response get shifted out into the MISR. At this point, the seed has been loaded, and the scan architecture is put into decompression mode to run the LFSR for m-q cycles. During those cycles, the remainder of the output response gets shifted into the MISR as the next test vector is generated.

6.1.2 Separate LFSR

In some cases it may be preferable to have a separate LFSR. For some designs (e.g., intellectual property cores, hard macrocells, legacy designs, etc.), it may not be possible or desirable to make any changes to the existing scan architecture. Another case where a separate LFSR may be preferable is if pseudo-random patterns are going to be used. An integrated LFSR has its state overwritten during the capture cycle, so it cannot be used as a proper pseudo-random pattern source (unless a circular BIST [Krasniewski 89], [Stroud 88], methodology is adopted). By having a separate LFSR, a mixed-mode BIST strategy can be employed in which pseudo-random patterns are first applied to detect the random-pattern-testable faults, and then LFSR reseeding is used to generate deterministic patterns to detect the random-pattern-resistant faults. Mixed-mode BIST can provide an even greater reduction in tester storage requirements.

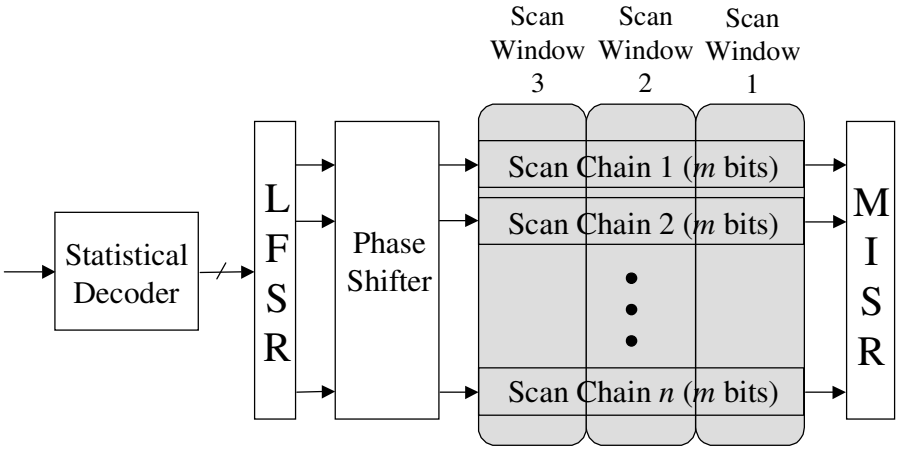


Figure 6.2. Architecture with Separate LFSR

One of the problems for all LFSR reseeding approaches where a separate LFSR is used is that the size of the LFSR scales with s_{max} and thus can become problematic for large industrial circuits. Here we propose a solution to this problem (a solution that can be applied for any LFSR reseeding scheme) which involves the use of what we will term "scan windows." The idea is to conceptually (not physically) partition the scan chains into scan windows, and use LFSR reseeding to fill each scan window one at a time, as illustrated in Fig. 6.2. In Fig. 6.2, the scan chains are divided into 3 scan windows where each scan window will have *n(m/3)* scan cells in it. So instead of generating an entire test cube with one LFSR seed, multiple seeds are used to generate a single test cube. By so doing, the number of specified bits that needs to be generated by each seed is reduced (i.e., the s_{max} for the scan windows is less than the s_{max} for the complete test cubes). The size of the scan windows can be chosen based on how large an LFSR is to be used. If an *r*-bit LFSR is to be used, then the size of the scan windows can be chosen so that the maximum number of specified bits for any scan window (i.e., the s_{max} of the scan windows) does not exceed *r*-20. Note that the LFSR can be any size provided an appropriate linear phase shifter is used [Rajski 98b].

LFSR reseeding with *w*-bit wide scan windows is performed by simply loading each seed into the LFSR and running it in autonomous mode for *w* cycles. After *m/w* scan windows have been loaded, then the scan chains contain a complete test cube. The

system clock is then applied and the process continues. Note that if the length of the scan chains do not divide evenly into scan windows, the scan vectors can be augmented with a sufficient number of X's so that the length of each scan vector is a multiple of the scan window size (during test application, the extra bits corresponding to the X's will simply be shifted off the end of the scan chain).

It is interesting to note that in the degenerate case where the scan window size is equal to 1, then the seed is essentially decompressed through a linear combinational network and loaded immediately into one bit-slice of the scan chains. This is effectively equivalent to what is proposed in [Bayraktaroglu 01] where they design a linear combinational network to expand a smaller input vector (which is essentially equivalent to a "seed") to fill one bit-slice of a larger number of scan chains. The advantage of having a larger scan window size is that the ratio of the total number of scan cells in the scan window versus the s_{max} of the scan window is generally much larger (it cannot be smaller), thus permitting a greater compression ratio (i.e., the number of scan bits that can be generated from the same size seed is greater). Moreover, in the method proposed here, we are providing even greater compression than simply doing a linear expansion of a seed because we are also statistically encoding the seed as well in a clever way. For a particular scan window, the number of specified bits may be much less than s_{max} and the method proposed here is able to take advantage of that to greatly improve the amount of compression. This is done by using the resulting degrees of freedom in the solution space of the linear equations to find seeds with short statistical encodings as will be described in Section 6.3.

Note that for both the architectures described above, once the LFSR is loaded with a seed, the LFSR then operates in an autonomous mode. Moreover, since the seeds are being compressed using a statistical code, different seeds might require different amounts of test data. Thus, this scheme can be classified as a periodically-loaded scheme using a variable number of bits per test cube.

6.2 Statistical Coding

In statistical coding, variable length codewords are used to represent fixed-length blocks of data. For example, if a data set is divided into 4-bit blocks, then there are 2^4 or 16 unique 4-bit blocks. Each of the 16 possible 4-bit blocks can be represented by a binary *codeword*. The size of each codeword is variable (it need not be 4 bits). The idea is to make the codewords that occur most frequently have a smaller number of bits, and those that occur least frequently to have a larger number of bits. This minimizes the average length of a codeword. The goal is to obtain a coded representation of the original data set that has the smallest number of bits.

A Huffman code [Huffman 52] is an optimal statistical code that is proven to provide the shortest average codeword length among all uniquely decodable variable length codes. An important property of Huffman codes is that they are prefix-free. No codeword is a prefix of another codeword. This greatly simplifies the decoding process.

The amount of compression that can be achieved with statistical coding depends on how skewed the frequency of occurrence is for the different codewords. If all of the codewords occur with equal frequency, then no compression can be achieved.

The idea proposed here is to encode the LFSR seeds using a statistical code. For an *r*-bit LFSR, the *r*-bit seed is partitioned into equal size blocks (if *r* does not divide out evenly, the length of the seed can be slightly extended by appending don't cares to make all the blocks equal size). The frequency of occurrence of the codewords can be skewed by intelligent selection of the seeds from the solution space of the linear equations as will be described in Section 6.3. Given the frequency of each codeword, an optimal Huffman code can be constructed [Huffman 52]. A decoder for the Huffman code can be implemented with a finite state machine (FSM) having 2^b-1 states for a *b*-bit block size. If the block size is chosen to be small (e.g. 4 or 5 bits blocks), then the Huffman decoder requires very little overhead. Alternatively, larger block sizes can be used with a selective coding approach to simplify the decoder (see [Jas 99] for details).

The Huffman decoder generates data one block at a time. For a separate LFSR, the flip-flops in the LFSR can have a parallel load capability and be loaded one block at a

time (as illustrated in Fig. 6.3). The assignment of LFSR flip-flops to blocks can be done in any manner (the blocks need not be contiguous). This degree of freedom can be used to improve the encoding efficiency as will be described in Sec. 6.3. For an integrated LFSR, the scan cells that make up the LFSR must be loaded by shifting the scan chains (they cannot have parallel load since that would overwrite the output response that they contain). For *b*-bit blocks, each block is loaded by shifting data into *b* scan chains in parallel. This is illustrated in Fig. 6.4 where there are 4-bit blocks which are loaded by shifting into 4 scan chains simultaneously. Note that if there are *n* scan chains, *n/b* blocks must first be decoded by the decoder, then all *n/b* blocks are shifted into one bit-slice of the scan chains during one scan cycle. So for the example in Fig. 6.4, the decoder would first decode two 4-bit blocks, and then shift them into one bit-slice of the 8 scan chains during one scan cycle. For the integrated LFSR, the assignment of LFSR flip-flops to blocks is much more constrained than for the separate LFSR. This must be taken into account when selecting seeds to skew the codeword frequency as will be described in Sec. 6.3.

For more details on implementing the statistical decoder and its interaction with the tester, the interested reader can see [Jas 99].

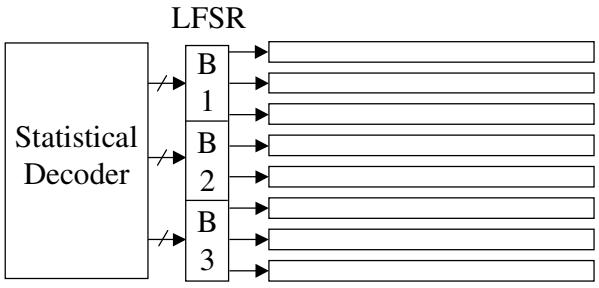


Figure 6.3. Division of Separate LFSR into Blocks (*B1-B3*)

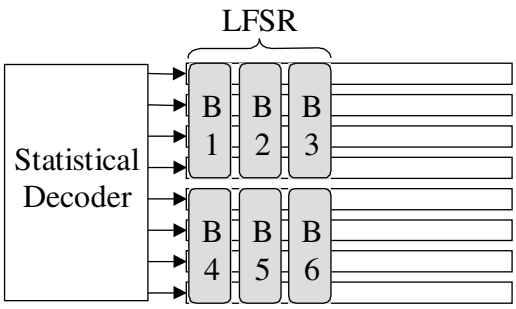


Figure 6.4. Division of Integrated LFSR into Blocks (*B1-B6*)

6.3 Using Linear Solution Space to Statistically Encode Seeds

Given the scan architecture, LFSR feedback polynomial, and phase shifter (if any), the LFSR can be symbolically simulated to determine the set of linear equations for each scan cell. This is done by having each bit in the seed of the LFSR be represented by a variable, and then the operation of the LFSR and phase shifter are symbolically simulated. Each time two bits are exclusive-ORed together, they form a linear combination of the corresponding variables. When the scan chains have been fully loaded, the final linear combination of variables that exists for each scan cell forms the linear equation for that scan cell. The symbolic simulation need only be done once.

For each test cube in the test set, a system of linear equations is formed in which there is one linear equation corresponding to each specified bit in the test cube. The resulting system of linear equations has the form $\boldsymbol{A}x=y$ where y is a column vector of the specified bits in the test cube (which can have up to s_{max} rows), and the solution x is the seed (which will have at least s_{max}+20 rows). In general there will be many solutions for x (i.e., many seeds that will produce the test cube). An example of a system of linear equations for a test cube is shown in Fig. 6.5, there is one row (i.e., equation) for each specified bit in the test cube. Gauss-Jordan Elimination [Cullen 97] can be used to transform a set of columns into an identity matrix (these columns are called the *pivots*) while the remaining columns are *free-variables*. Figure 6.6 shows the resulting matrix after Gauss-Jordan Elimination has been performed on the example. Without loss of generality, suppose the first p columns are pivots and the remaining f columns are free-

variables, then the set of solutions for x can be represented as any combination of values for the free-variables (x_{p+1} to x_{p+f}), and then the pivots (x_1 to x_p) are calculated as the linear sum of the column vectors for the free-variables with value 1. The set of solutions for the example is shown in Fig. 6.7.

Our goal is to choose a seed solution for each test cube such that the complete set of seeds for all the test cubes can be efficiently encoded using a statistical code. To increase the encoding efficiency, the frequency of occurrence for the codewords should be as skewed as possible. To accomplish this, we use a two phase process first doing global processing across all seeds to select the statistical encoding, and then optionally doing local processing one seed at a time to fine tune the result.

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Figure 6.5. Example of System of Linear Equations

$$\begin{array}{c} \overbrace{}^{\text{Pivots}}\ \overbrace{}^{\text{Free Variables}} \\ \left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 1\ 1\ 1 & 1 \\ 0 & 1 & 0 & 0 & 0\ 1\ 0 & 0 \\ 0 & 0 & 1 & 0 & 0\ 1\ 1 & 1 \\ 0 & 0 & 0 & 1 & 0\ 1\ 0 & 0 \end{array}\right] \end{array}$$

Figure 6.6. Gauss-Jordan Reduction of Example in Fig. 6.5

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = x_5 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + x_6 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + x_7 \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Figure 6.7. Solution Space for Example in Fig. 6.5

6.3.1 Global Processing

To make the frequency of occurrence for the codewords as skewed as possible, the same bit positions in the seeds should have the same value as often as possible. Notice that in using Gauss-Jordan Elimination to find a solution for a particular seed, the free-variables can be assigned any value independently of each other (whereas the pivots are dependent on the values assigned to the free-variables). The key idea in our strategy is to do Gauss-Jordan Elimination when solving for each seed and to process the columns in the same order every time (e.g., first try to make x_1 a pivot, then try to make x_2 a pivot, etc.). Each seed has its own system of linear equations, if we order the columns in the $\boldsymbol{A}$ matrix for each seed in the same way, and then process the columns from left to right when performing the steps of Gauss-Jordan Elimination, then we have observed that the probability of a column ending up as a pivot decreases significantly from left to right. Some columns in $\boldsymbol{A}$ may be linearly dependent and become all 0's during Gauss-Jordan Elimination thus moving the final pivot toward the right, however, if the LFSR has a primitive feedback polynomial and an appropriate phase shifter is used, then probability of linear dependence in the columns is fairly low. Thus for a test cube with s specified bits, the final pivot will tend to be fairly close to the s-th column. Since we can set the value of the free-variables to anything, we can begin by setting them all to value 0 (the value of the pivots are dependent on the value of the free-variables, so they will take on a particular set of values based on this assignment to the free-variables). Since the columns to the right are skewed towards free-variables which are all set to 0, the corresponding bit positions in the seeds will be skewed to 0 which is good for statistical coding.

An example of the tendency for the pivots to be towards the left is shown in Fig. 6.8. The graph shows the percentage of seeds that have a pivot in each column from left to right when solving the system of linear equations for the test cubes for circuit *s9234*. As can be seen, the pivots are very much skewed to the left, and thus the bits towards the right of the seed will be very skewed towards 0.

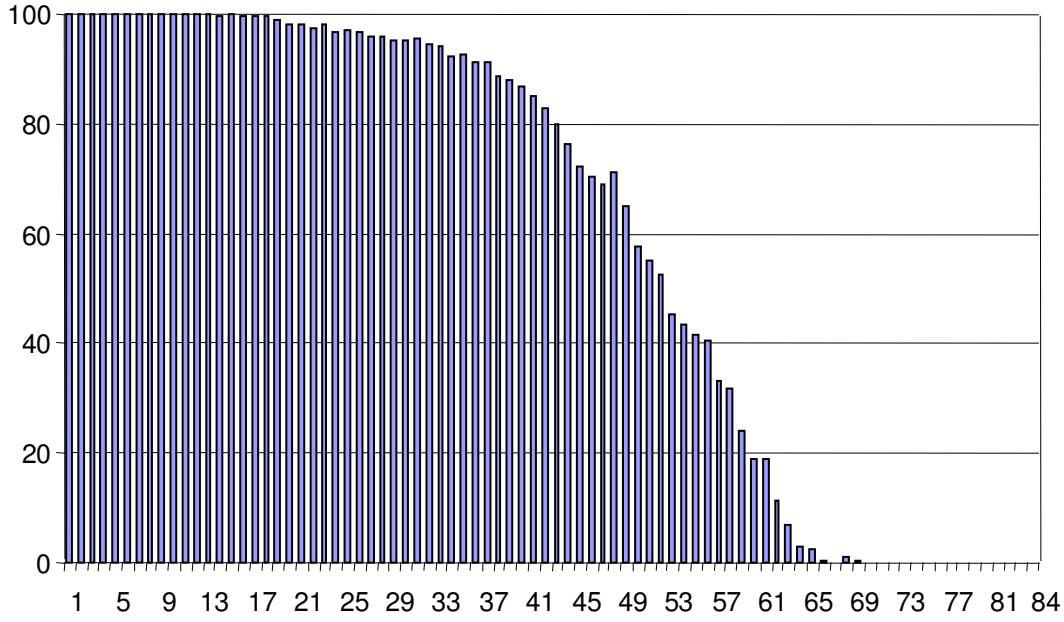


Figure 6.8. Percentage of Seeds Having Pivot in Each Column
(Gauss-Jordan Elimination Ordered from Left to Right)

For a separate LFSR, the assignment of LFSR flip-flops to blocks for statistical encoding can be done in any manner. Thus, we can compute the percentage of 0's in each bit position of the set of seeds and sort them. If there are k blocks in each seed, then the k bit positions in the set of seeds that are most skewed towards 0 can be assigned to the most significant bit position in each of the k blocks. The next k bit positions that are most skewed towards 0 can be assigned to the second most significant bit position in each of the k blocks, and so forth. Lastly, the k bit positions that are least skewed toward 0 (and therefore most skewed towards 1) are assigned to the least significant bit positions of the k blocks. At this point, each seed has been mapped to k blocks where the number of 0's and 1's in each bit position of the blocks have been skewed. This causes the frequency of occurrence of the codewords to be skewed, and thus allows the statistical coding to compress the data.

For an integrated LFSR, the assignment of LFSR flip-flops to blocks is significantly constrained (as was described in Sec. 6.2). However, we can compensate for this by using the degree of freedom in setting the ordering of the Gauss-Jordan Elimination to

still get nearly the same effect. In this case, we make the assignment of bit positions in the seeds to blocks before solving the linear equations. We can choose this assignment to simplify the process of loading the blocks from the decoder to the scan chains. Then based on where each bit position in the seeds occurs in the blocks, we can set the order in which the Gauss Jordan Elimination is done for all the seeds. If there are k blocks in each seed, then the k bit positions in the seeds that correspond to the most significant bit position in the blocks can be ordered on the far right side of the $\boldsymbol{A}$ matrix. Those corresponding to the second most significant bit position can be ordered just to the left of that, and so forth. Lastly, those corresponding to the least significant bit position are ordered on the far left side of the $\boldsymbol{A}$ matrix. The reason for doing this is that when the Gauss Jordan Elimination proceeds from the left to the right, the columns towards the right will tend to have the most free-variables and thus tend to be most skewed. The most skewed columns will be aligned in the blocks to again cause the frequency of occurrence of the codewords to be skewed.

So far we have only discussed the case where all the free-variables are set to 0. It is also possible to consider solutions where some or all of the free-variables are set to 1 as well. Free-variables that correspond to the same bit position in the blocks should all be set to the same value (either 0 or 1) to ensure the best frequency skew. Within this constraint, however, other assignments to the free-variables can be explored for different bit positions in the blocks. Changing the assignment of the free-variables will change the values of the pivots, and may give a slightly better frequency skew. An optional step would be to try several different assignments of the free-variables for different bit positions of the blocks and keep the one that gives the best frequency skew in the blocks.

Once the final assignment of free-variables has been selected and the bit positions in the seeds have been assigned to blocks, then the frequency of each codeword can be calculated. Based on this frequency distribution, the optimal Huffman code can be constructed by forming a Huffman Tree [Huffman 52], or a selective statistical code can be formed [Jas 99].

6.3.2 Local Processing (Optional)

Given the statistical code that is to be used, the number of bits required to encode each block is known. An optional step is to do some local processing on one seed at a time to fine tune the result. The system of linear equations for each test cube is independent of all the other test cubes, so changing the assignment of the free-variable for one seed has only a local effect on the pivots of that seed. A different assignment of the free-variables for a seed (different from the global assignment which was done across all seeds) may result in a shorter encoding for the seed. For each seed, a branch and bound procedure can be used to try to find a better assignment of the free-variables. Changing the value of one free-variable will change the codeword for the block in which the free-variable resides (likely making it worse, i.e., making the encoding longer), however it may also change the value of several pivots thereby changing the encoding of the blocks in which they reside (possibly making them better, i.e., making the encoding shorter). The net effect may be to reduce the number of bits needed to encode the seed as a whole. A branch and bound procedure can be used to search for a better assignment of the free-variables. The bound is formed by the best encoding that has been seen so far. The search space is exponential in the number of free-variables in the worst-case, so it may be necessary to place some limit on the maximum number of iterations used for each seed. When all seeds have been processed, then the new frequency of each codeword is calculated and the final statistical code is computed.

6.4 Experimental Results

Experiments were performed on the largest ISCAS 89 benchmark circuits [Brglez 89]. For each circuit, ATPG was performed to generate test cubes for the non-redundant faults. The system of linear equations for each test cube was then processed with Gauss-Jordan Elimination always following the same ordering as described in Sec. 6.3. The two phase process of global and local processing were then done to obtain a set of seeds for all the test cubes that could be efficiently encoded using a statistical code. Once the seeds were obtained, they were encoded using a Huffman code, and the total

test storage requirement was then computed. The results are presented in Table 6.1. The test storage requirement is the number of encoded bits that would have to be stored on the tester. The percentage compression shows the compression achieved by storing the encoded seeds on the tester as compared with the test storage requirements for the unencoded test vectors (i.e., simply storing the test vectors themselves on the tester). As can be seen, the compression obtained by using the integrated LFSR technique is very high. Results are also shown for the separate LFSR architecture for different scan window sizes followed by the size of the LFSR used in each case (the LFSR size varies because the maximum number of specified bits, s_{max}, for different scan window sizes varies). In comparing the results for using an integrated LFSR versus a separate LSFR, it can be seen that the integrated LFSR provides better results as is expected since it is using a larger LFSR and is generating the entire test cube with each seed.

Table 6.2 compares the proposed method (using an integrated LFSR with a block size of 4) with previously published test vector compression schemes as well as with MINTEST [Hamzaoglu 98], which is an ATPG procedure that uses dynamic compaction. In this case, the test sets for the different schemes are not the same, so the optimality of the ATPG and compaction procedures used to obtain the test sets do impact the results. For the results shown for statistical coding of the test cubes, we used the methodology described in [Jas 99] on the same set of test cubes that were used for the proposed method (i.e., the numbers are not being taken directly out of the [Jas 99] paper itself). This shows a direct comparison between encoding the test cubes themselves with a statistical code versus encoding LFSR seeds that produce the test cubes as is proposed here. As can be seen, much greater encoding efficiency can be obtained by encoding the seeds.

Table 6.1. Results for Integrated and Separate LFSR for Block Size of 4

Circuit			Integrated LFSR			Separate LFSR			
Name	Scan Elements	Num. Test Cubes	LFSR Size	Test Storage (bits)	Percent Compress	Scan Window Size	LFSR Size	Test Storage (bits)	Percent Compress
s5378	214	196	40	6,180	85	107 54 27	28 20 12	6,667 7,306 8,028	84 83 81
s9234	247	205	68	12,112	76	124 62 31	38 24 24	12,445 13,059 12,955	75 74 74
s13207	700	266	48	11,285	94	350 175 88	36 20 16	11,859 12,079 14,079	94 94 92
s15850	611	269	54	12,438	92	306 153 77	32 24 20	12,663 14,439 17,033	92 91 90
s38417	1664	376	104	34,767	94	832 416 208	60 38 28	36,430 37,773 42,360	94 94 93
s38584	1464	296	120	29,397	93	732 366 183	61 38 28	30,355 31,971 35,427	93 93 92

Table 6.2. Comparison of Test Data for Different Encoding Schemes

Circuit Name	MinTest [Hamzaoglu 98]		Illinois Scan Architecture [Hamzaoglu 99]		FDR Codes [Chandra 01]		Linear Decompressors [Bayraktaroglu 01]		Statistical Coding of Test Cubes [Jas 99]		Proposed Approach	
	Num. Vect.	Total Bits	Num. Vect.	Total Bits	Num. Vect.	Total Bits	Num. Vect.	Total Storage	Num. Vect.	Total Bits	Num. Vect.	Total Bits
s5378	97	20,758	160	14,572	111	12,346	NA	NA	196	15,417	196	6,180
s9234	106	26,182	238	27,111	159	22,152	NA	NA	205	19,912	205	12,112
s13207	233	163,100	273	109,772	236	30,880	251	24,096	266	52,741	266	11,285
s15850	96	58,656	178	32,758	126	26,000	170	16,320	269	49,163	269	12,438
s38417	68	113,152	337	96,269	99	93,466	296	63,936	376	172,216	376	34,767
s38584	110	161,040	239	96,056	136	77,812	182	34,944	296	128,046	296	29,397

Chapter 7

Adjustable Width Linear Combinational Scan Vector Decompression

The earlier chapters described three periodically-loaded linear expansion schemes. While these schemes provide very good results as seen from the experimental results, they are also well suited for a system-on-chip (SOC) environment where multiple cores can be tested concurrently so that the available tester bandwidth is fully utilized. This chapter describes a continuous-flow technique that can efficiently utilize the tester bandwidth since it keeps receiving data from the tester every clock cycle while concurrently shifting data into the scan chains. Another nice feature of continuous-flow techniques is that the test data can be applied in a manner similar to conventional ATPG test vectors. This keeps things simple from a tool integration standpoint.

Linear expansion schemes can be divided into those that use sequential linear finite state machines and those that use only linear combinational circuits. The schemes that use linear finite state machines include virtual scan chains [Jas 00], LFSR reseeding [Könemann 91], SmartBIST [Könemann 01], partial reseeding [Krishna 01], seed compression [Krishna 02], Embedded Deterministic Test [Rajski 02], and efficient seed utilization for reseeding [Volkerink 03]. The schemes that use only linear combinational circuits may provide less compression, but they require less overhead and are simpler and easier to implement. This chapter describes a combinational linear expansion scheme (this work has been published in [Krishna 03b]) while the next chapter describes a scheme that uses a sequential linear expansion circuit.

Note that the term linear decompression can be used interchangeably in place of linear expansion since the linear expansion process basically involves decompressing the encoded data stored on the tester by using the linear circuit on the chip. Hence linear expansion circuits can also be called linear decompressors.

As illustrated in Fig. 7.1, the combinational linear decompression schemes involve placing a combinational circuit between the tester channels and the scan chains that

expands a small number of tester channels, *N*, to fill a much larger number of scan chains, *M*. Two basic schemes have been proposed for combinational linear decompression. The very simplest is the Illinois Scan Architecture [Hamzaoglu 99], [Hsu 01], in which one channel from the tester is used to feed multiple scan chains. Another scheme is the one described in [Bayraktaroglu 01] which involves using a network of XOR gates such that each scan chain is fed from some linear combination of the channels from the tester. The techniques in [Bayraktaroglu 03] and [Mitra 03] also use a combinational circuit consisting of XOR gates.

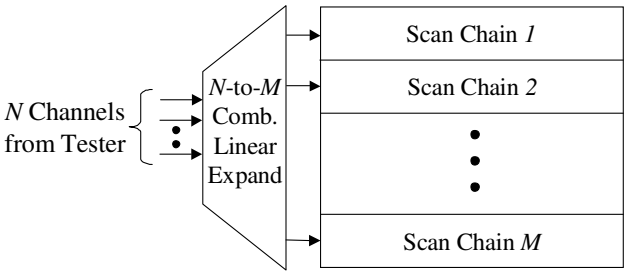


Figure 7.1. Combinational Linear Decompression

In this chapter, a new scheme is proposed for combinational linear expansion that has some nice advantages. One of the drawbacks of the schemes in [Bayraktaroglu 01, 03] and [Mitra 03] is that in every clock cycle, an entire "bit-slice" of the scan chains (which consists of M bits) must be filled. Thus, the most specified bit-slice places an upper bound on the compression ratio of *M*/*N*. In order to keep the compression ratio high, some constraints need to be placed on the ATPG process so that no bit-slice has too many specified bits such that it cannot be generated through the linear expansion network. Placing constraints on the ATPG process can result in generating more test vectors thereby offsetting some of reduction in test data gained through using compression. In the proposed scheme, an adjustable width linear expansion technique is used. Whereas the schemes in [Bayraktaroglu 01, 03] and [Mitra 03] always use a fixed expansion width of *N* bits to fill *M* scan chains in every clock cycle, the proposed approach provides the capability of using different size expansions. In any given clock cycle, the *N* bits coming from the tester may expand to load all *M* scan chains, or may load only a subset of the

scan chains. This provides some nice benefits. It eliminates the requirement that every scan bit-slice has to be in the output space of the linear decompressor. The decompressor can be designed so that any scan vector can be generated using the proposed scheme regardless of the number or distribution of the specified bits (as explained in Sec. 7.2). Thus, the proposed scheme allows the use of any ATPG procedure without any constraints. Moreover, it allows greater compression to be achieved. The ratio of M/N can be scaled much larger. For lightly specified bit-slices, the proposed scheme can use a full N to M expansion, while for more heavily specified bit-slices it can use a smaller width expansion. This flexibility allows for better encoding efficiency and hence more compression.

7.1 Overview of Proposed Scheme

The key idea of the proposed scheme is that the N bits coming from the tester each clock cycle need not load exactly one complete bit-slice of the scan chains. Based on the number of specified bits in the bit-slice, the N bits can be used to encode either a portion of the bit-slice (for a bit-slice having a high number of specified bits) or an entire bit-slice (for low number of specified bits). Hence the bit-slice with the maximum number of specified bits does not place a lower bound on the number of tester channels required, since multiple tester cycles can be used to load the bit-slice. No special ATPG is thus required to distribute the specified bits evenly across the bit-slices.

The architecture for the proposed scheme is shown in Fig. 7.2. The N bits from the tester are connected to a combinational linear decompressor. The outputs of the decompressor feed the scan chains within the circuit. If there are M scan chains, the linear decompressor has M outputs. Each bit-slice of the scan chains is divided into k equal sized blocks, B_1 through B_k , where each block has q bits. Thus, the number of blocks, k, is equal to M/q. The scan chains are effectively partitioned into k groups, corresponding to the blocks B_1 though B_k. Each of these scan groups has a separate scan clock associated with it. As can be seen from Fig. 7.2, the first q scan chains corresponding to block B_1 are clocked by scan clock SC_1, and the last q scan chains

corresponding to block B_k are clocked by scan clock SC_k. These scan clocks are generated by the control logic shown in Fig. 7.2.

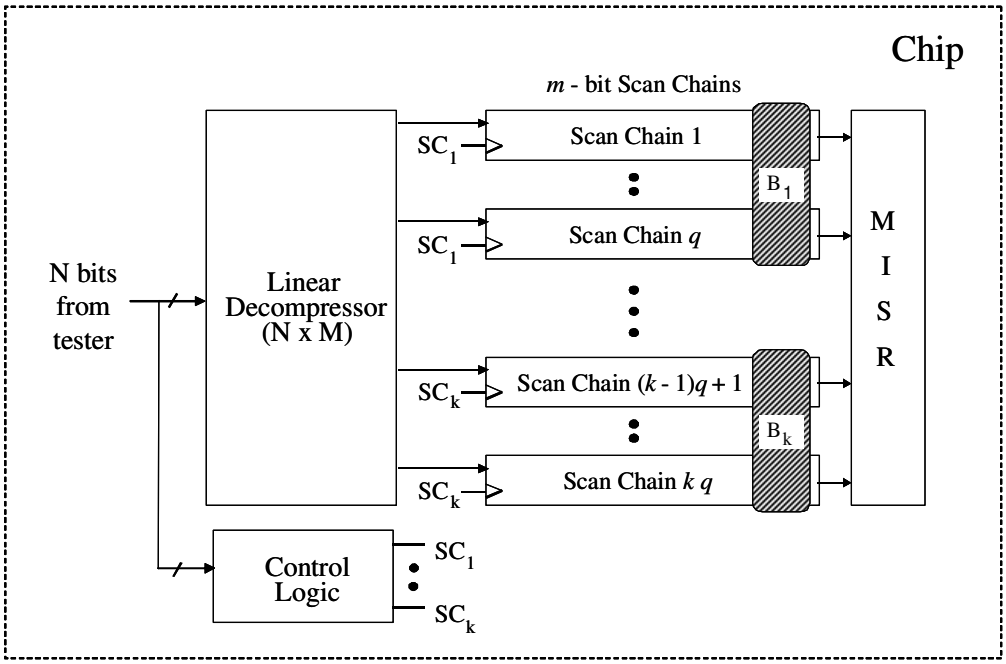


Figure 7.2. Architecture for Proposed Scheme

Using this architecture, an adjustable width linear expansion can be implemented. The N bits coming from the tester can be used to load either just a single block (by activating only the scan clock for that block) or multiple blocks. There would be no compression achieved when N is used to load just a single block (this would occur when the block to be encoded has a very high number of specified bits). However, the compression would be high when the N bits are used to load a large number of blocks at a time (this would occur when successive blocks have very few specified bits).

The number of blocks that is loaded in a clock cycle can be termed as the grouping factor g. Some control information is needed to select the grouping factor that is used in each clock cycle. For example, if g can take the values 1, 2, 4 and 16, then two control bits would be required to encode these four possible values of g. To minimize the control

information, g should be assigned only a limited number of values. The number of control bits, p, required to select the grouping factor, g, for each clock cycle is given by:

$p = \lceil \log_2(\text{ \# of possible values of } g\ ) \rceil$

The N bits coming from the tester each clock cycle must encode the p control bits for the grouping factor and the corresponding blocks of the bit-slice. The control logic shown in Fig. 7.2 decodes the p control bits for the grouping factor from the N bits coming from the tester. The control logic also keeps track of the block that was loaded last during the previous clock cycle, since the blocks to be loaded during the current cycle will start after this block. The control logic thus needs a $\lceil \log_2(k) \rceil$ bit index register that points to the last block that was loaded in the previous clock cycle. Based on the value of this index register and the grouping factor, the control logic selects the set of scan clocks that need to be activated in the current clock cycle. The index register is updated in the current cycle by adding the grouping factor to the contents of the register by performing a modulo-k addition so that it points to the last block that was loaded.

Consider the operation of the hardware in Fig. 7.2. During each clock cycle, the tester feeds N bits to the linear decompressor and the control logic. Since the linear decompressor is a combinational network, the M outputs of the decompressor are ready with the decoded data within the same cycle. Based on the N bits from the tester, the control logic activates the appropriate set of scan clocks. The remaining scan clocks are left inactive. For example, if block B_1 was loaded last during the previous clock cycle, and the grouping factor for the current clock cycle is 2, then blocks B_2 and B_3 are loaded during the current clock cycle. Only the scan clocks SC_2 and SC_3 are activated during the current clock cycle. Note that blocks across different bit-slices can also be loaded within the same clock cycle. If block B_{k-1} was the last block to be loaded in the previous clock cycle and the grouping factor is 3, then in the current clock cycle, the scan clocks, SC_1, SC_2, and SC_k will be activated. This will cause block B_k from the first bit-slice and blocks B_1 and B_2 from the second bit-slice to be loaded. Thus, during each clock cycle, at most k blocks can be loaded, though not necessarily from the same bit-slice (as it may

wrap to the next bit-slice). This process continues until all the bit-slices are loaded. At this point, the system clock can be applied and the response from the circuit loaded into the scan chains. Note that no wrapping is done across scan vectors.

Assuming that the values of the grouping factor range from just a single block to all the k blocks within a bit-slice, the number of clock cycles required to load two test cubes could vary by a factor of k times. Thus depending on the specified bit distribution, each test cube could require a different number of clock cycles, leading to a difference in the number of bits stored on the tester for each test cube. Moreover, since the chip receives data from the tester every clock cycle, this scheme falls under the category of continuous-flow schemes using a variable number of bits per test cube.

A multiple input signature register (MISR) can be used to compress the output response as it is shifted out of the scan chains while the next scan vector is being loaded. In conventional scan testing, the output response stored in an entire bit-slice is shifted into the MISR during each clock cycle. However, note that in the proposed scheme, when not all the blocks in a bit-slice are loaded in a single clock cycle, only a subset of the scan chains are shifted in that clock cycle. The scan-out for the inactive scan chains will simply hold the same value as the last clock cycle. This does not cause any problems for the MISR operation as it will simply compact the scan-out of all the scan chains each clock cycle regardless of whether each scan chain is active or not.

7.2 Design of Linear Combinational Decompressor

It is important that the linear combinational decompressor be designed in such a way that it is always possible to encode all scan vectors regardless of the number and distribution of the specified bits. This eliminates the need for any special constraints on the ATPG process. To accomplish this, the linear combinations of the channels coming from the tester that are used to drive each scan chain need to be carefully selected to ensure that it is always possible to solve for at least one block in each clock cycle.

The linear decompressor consists of XOR gates driving the scan-ins of each of the M scan chains (one XOR gate per scan chain). The inputs to each XOR gate are a subset of

the N channels from the tester. Thus, each scan chain is driven by some linear combination of the N channels from the tester. The key is to choose the linear combinations for each scan chain in such a way that it is always possible to solve for at least one block. To accomplish this, the parameters for the architecture of the proposed scheme should be selected so that $N = p+q$, where p is the number of control bits and q is the block size. Then the following procedure can be used for selecting the linear combinations. First select the linear combinations for the p control bits so that they are linearly independent of each other. This ensures that any combination of the control bits can be generated each clock cycle. Then for each block of q scan chains, the linear combinations for each scan chain are chosen so that they are linearly independent of each other and linearly independent of the linear combinations of the p control bits. This ensures that any combination of specified bits for the block can be generated. Forming the $p+q$ linearly independent combinations is always possible provided N is not less than $p+q$. By increasing the number of inputs to the XOR gates, the space of possible linear combinations increases (it is maximum for $N/2$). This allows more diversity in the linear combinations across different blocks; however, it comes at the cost of more overhead. As shown in [Bayraktaroglu 01], good results can typically be obtained using just 3-input XOR gates.

For the proposed decompression scheme, each scan vector is encoded by starting from the first block in the first bit-slice. The largest grouping factor is tried first. For each specified bit in the set of consecutive blocks corresponding to the largest grouping factor, a linear equation is formed corresponding to the XOR gate that is driving its scan-in. A linear equation is also formed for each of the p control bits so that it takes on the correct value corresponding to the largest grouping factor. The system of linear equations for all the specified bits in the set of blocks and the p control bits is then passed to a linear solver. If a solution cannot be found, then the next smaller grouping factor is tried and so forth. In the worst case, it may be necessary to go all the way down to the smallest grouping factor which will correspond to just a single block. This is guaranteed to have a solution since all the linear equations of a block will be independent due to the way the linear decompressor was designed. When the linear solver does find a solution,

then the solution gives the values for the N bits coming in from the tester for that clock cycle, and the encoding process then continues for the next set of blocks in the same manner. This process continues until all the blocks of the scan vector have been encoded. A more detailed description of how to form and solve the linear equations can be found in [Könemann 91] and [Krishna 01].

7.3 Selecting Architecture Parameters

Given the set of test cubes (test vectors where the unspecified inputs are left as don't cares) and the tester bandwidth available, this section describes how to select the parameters for the proposed scheme.

N is fixed by the available tester bandwidth, since N is the number of bits coming from the tester every tester cycle. The designer can select whatever number, M, of scan chains is desired. The amount of compression will improve as the number of scan chains is increased, but the overhead of the decompressor increases as well and there are diminishing marginal returns as the number of scan chains becomes large. Note that there may be other design factors that also impact how many scan chains the designer wishes to use.

So given a fixed N and M, the remaining parameters to select are the number of control bits, p, and the corresponding set of grouping factors. Once p is chosen, then the size of each block, q, is equal to N-p, and the number of blocks (k) per scan bit-slice is equal to M/q.

As the number of control bits is increased, more grouping factors are possible which improves encoding efficiency. However, this is offset by the additional data required for the control bits. At some point, the marginal gains in encoding efficiency do not compensate for the additional data required for adding another control bit. Figure 7.3 shows an example of how the amount of compressed test data varies with the number of control bits for the s13207 benchmark circuit. As can be seen, it reaches a minimum when p=3 and increases thereafter.

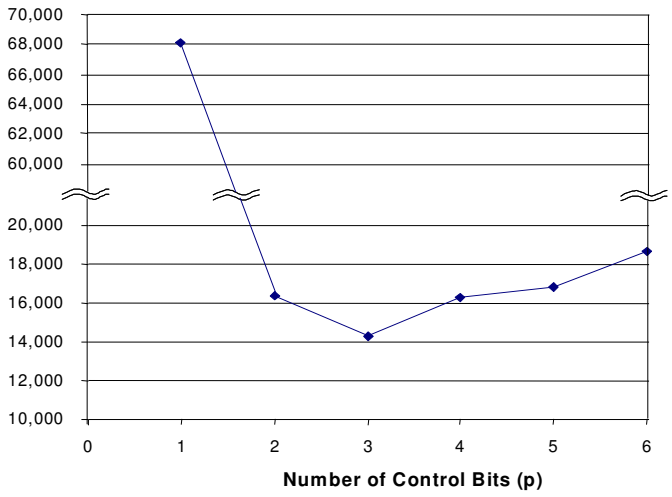


Figure 7.3. Total Compressed Data versus Number of Control Bits for s13207 Benchmark Circuit

So the procedure for selecting the number of control bits is as follows. First try to encode all the scan vectors using full N to M expansion which corresponds p=0, i.e., having no control bits (which is the same as [Bayraktaroglu 01]). If this is possible, then that is the best solution for those values of M and N. If it is not possible, then try p=1. This allows 2 different grouping factors. One can be a grouping factor of 1 and the other a grouping factor of k. The test cubes can be encoded using these two grouping factors to compute the total amount of compressed test data. Then try p=2. This allows 4 different grouping factors. For simplicity, they can be selected to be equally spaced between 1 and k. Again the total amount of compressed test data can be calculated. This process continues with p increasing by one each time until a point is reached where increasing p does not reduce the total test data. This procedure allows the minimum point in the graph to be found thereby giving the best value for p to minimize the total amount of compressed test data.

One simplification in the procedure above is that the grouping factors are chosen to be equally spaced between 1 and k. This may not give the most efficient set of grouping factors. An optimization for the procedure would be to first assume all grouping factors are possible, and then encode the scan vectors and keep track of how often each grouping

factor is used. Given the frequency with which each grouping factor is used, a more optimal set of grouping factors can be chosen.

The proposed scheme for adjustable width linear combinational expansion is simple to implement, yet it efficiently exploits the don't care values in the test set to reduce test data storage requirements. The overhead for the proposed scheme is small. It consists of one 3-input XOR gate for each scan chain, a small amount of control logic, and an index register. The computational complexity for solving the linear equations is $O(N^2M)$ where N is the number of channels from the tester and M is the number of scan chains. Hence, it can be done very rapidly. Thus, the proposed scheme is very practical and provides a low cost solution for achieving an order of magnitude reduction in test data volume.

7.4 Experimental Results

Experiments were performed on the largest ISCAS 89 benchmark circuits. For each circuit, ATPG was performed to generate test cubes for the non-redundant faults. The test cubes were then encoded using the proposed scheme. The results are shown in Table 7.1. The number of test cubes for each circuit is shown. The set of test cubes was encoded using the proposed scheme assuming 19 channels coming from the tester. Results are shown for using 1, 2, and 3 control bits. In each case, the total amount of required tester storage (including the control bits) is shown followed by the percent compression.

Table 7.2 provides a comparison between the proposed method and the linear combinational decompressor scheme in [Bayraktaroglu 01] which uses a fixed width expansion. While the proposed scheme does not require any special ATPG as it is guaranteed to work for any set of test cubes, the scheme in [Bayraktaroglu 01] requires a special constrained ATPG to ensure all test cubes can be encoded. Note that if the proposed method was combined with a special ATPG procedure, the results could be improved significantly. Nonetheless, the proposed scheme performs very well as the adjustable width expansion allows better encoding efficiency compared with a fixed width expansion even without any modifications to the ATPG process.

Table 7.1. Results using 1, 2, and 3 Control Bits

Circuit		$p = 1$		$p = 2$		$p = 3$	
Name	Num Cubes	Test Data	% Compression	Test Data	% Compression	Test Data	% Compression
s13207	266	68,134	63.4	16,378	91.2	**14,307**	92.3
s15850	226	85,823	37.8	22,895	83.4	**15,067**	89.0
s38417	376	197,334	68.4	51,623	91.7	**49,001**	92.1
s38584	296	197,182	54.4	42,997	90.0	**28,994**	93.3

Table 7.2. Comparison with Scheme Described in [Bayraktaroglu 01]

Circuit Name	[Bayraktaroglu 01]			Proposed Scheme		
	Tester Channels	Num Cubes	Test Data	Tester Channels	Num Cubes	Test Data
s13207	24	251	24,096	19	266	14,307
s15850	32	170	16,320	19	226	15,067
s38417	32	296	63,936	19	376	49,001
s38584	24	182	34,944	19	296	28,994

Chapter 8

3-Stage Variable Length Continuous-Flow Scan Vector Decompression Scheme

The previous chapter described a continuous-flow technique that uses a combinational linear expansion network. This chapter also describes a continuous-flow technique, but one which uses a sequential linear expansion network (this work was published in [Krishna 04]). The proposed scheme has a number of nice features. It uses a continuous-flow decompressor so it can be very efficiently connected directly to the tester. The architecture for the decompressor combines three different stages of linear expansion such that it is very efficient for any distribution of specified bits in a test cube. It can vary the number of bits that are used to encode each pattern (i.e., it does not use a fixed number of bits to encode every pattern). Thus, it can efficiently compress any test cube regardless of the number of specified bits. As a result of this feature, there is no need for any constraints on the automatic test pattern generation process. Any ATPG can be used with any amount of static or dynamic compaction. It has important applications for situations where it is not possible or desirable to use a special ATPG that is tailored for the decompressor. For example, if an existing set of test cubes is available, or for circuits where unconstrained static and dynamic compaction can significantly reduce the total number of vectors. The proposed scheme requires low hardware overhead as it configures the decompressor out of the scan cells themselves.

The first continuous flow scheme that used a variable number of free-variables per test cube was part of the SmartBIST family of decoders described in [Könemann 01]. This method involves injecting free-variables from the tester into a set of LFSRs that collectively have as many stages as the number of scan chains being driven. One channel from the tester is used for controlling a clock gating mechanism that controls when the scan chains are loaded from the set of LFSRs receiving the free-variables. When a sufficient number of free-variables have been injected into the LFSRs to solve the linear equations for the specified bits in the next bit-slice of the scan chains, then the clock

gating mechanism allows the scan chains to be loaded. Most of the time, the linear equations will be solved immediately, so the clock gating mechanism is only used occasionally when a bit-slice has too many specified bits to solve right away. This scheme is capable of applying any test cube regardless of the number of specified bits it has. The clock gating control signal is controlling how many free-variables are used per bit-slice and hence per test cube. One nice feature of this scheme is that it has a very modular design. A drawback is that one tester channel needs to be completely devoted to providing the clock gating signal. A scheme based on static LFSR reseeding was described in [Volkerink 03] which uses the same control mechanism as [Könemann 01], but uses a large LFSR (e.g., 500 bits) along with a shadow register to allow it to decompress more than one bit-slice at a time. A scheme based on seed overlapping was described in [Rao 03] which is based on the method in [Bayraktaroglu 01] but allows seeds of different sizes to be used. It also decompresses one scan bit-slice at a time and requires a special constrained ATPG.

The proposed scheme is a continuous-flow scheme with a variable number of free-variables per test cube which combines three different stages of linear expansion. Unlike the previous methods, it does not require an extra tester channel for the control information, rather the control information is reduced to only a few bits per test cube thereby freeing up the tester channel so it can be used to inject more free-variables instead. Instead of solving for one scan bit-slice at a time as in [Könemann 01] and [Rao 03], the entire test cube is solved together which allows greater encoding efficiency. Whereas the method in [Volkerink 03] requires a large LFSR along with a shadow register and the number of bit-slices that it can solve for at a time is limited by the size of the LFSR, in the proposed approach, the decompressor is constructed from the scan elements themselves thereby reducing hardware requirements and allowing it to solve for the entire test cube. Unlike the method in [Rao 03], no special constrained ATPG is required for the proposed approach. The proposed 3-stage decompressor provides extremely high encoding efficiency for any distribution of specified bits.

8.1 Architecture for Proposed Scheme

The architecture for the proposed scheme is shown in Fig. 8.1. The portion shown in gray is configured from the scan cells themselves. In each clock cycle, a *b*-bit block of free-variables arrives from the tester. The *b*-bit block is expanded through a linear combinational expansion circuit similar to the method in [Bayraktaroglu 01, 03] to load *n* scan chains. Each of the *n* outputs of this linear combinational expansion circuit is generated through a three-input XOR gate which forms a linear combination of the free-variables. In [Bayraktaroglu 01, 03], constraints are placed on the ATPG to ensure that every bit-slice can always be generated (i.e., the linear equations can be solved) immediately through the combinational circuit. This is possible if the number of specified bits in each (*n*-bit) bit-slice of the scan chains is approximately less than *b* (note that it is possible to solve the equations for more specified bits than *b*, but the probability of success diminishes exponentially). While this makes things simple and is an attractive aspect of this approach, it becomes very restrictive as the ratio of *n* to *b* becomes large hence limiting the amount of static and dynamic compaction that can be done resulting in the ATPG generating more test cubes or possibly not being able to detect some faults under these restrictions. To avoid these limitations, and to get a much better encoding efficiency, the proposed method configures the scan cells into a set of LFSRs.

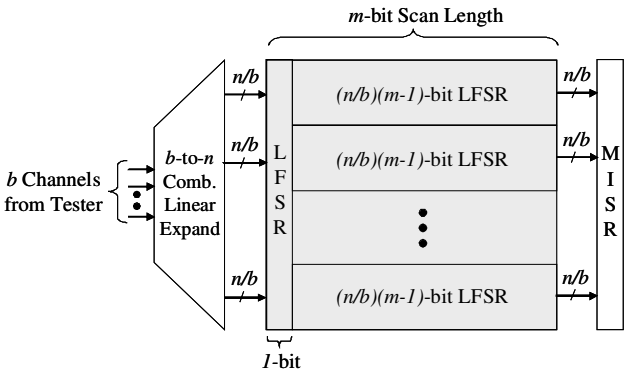


Figure 8.1. Architecture for Proposed Scheme for *n* Scan Chains

The idea of configuring an LFSR from the scan cells for LFSR reseeding was first described in [Zacharia 95, 96], [Rajski 98a, 99]. In the proposed scheme, a set of LFSRs is configured as shown in Fig. 8.1. There is a "small vertical LFSR" which in turn feeds a set of *b* "large horizontal LFSRs." The small vertical LFSR is one bit wide and *n* bits long. It is configured by having the output of each scan cell in the leftmost bit-slice of the scan chains be XORed with the scan-in of the next scan chain (the last scan chain wraps around to the first) and then including some feedback taps to implement a primitive polynomial for a modular (type II) LFSR. Each of the large horizontal LFSRs have *(n/b)(m-1)* stages where *m* is the scan length of each scan chain (if the scan chains are not balanced in length, then the size of each LFSR would vary accordingly). The large horizontal LFSRs are configured by partitioning the scan chains into *b* groups of *n/b* scan chains. The final scan-out of a scan chain in a group is XORed with the scan-in of the first scan cell of the next scan chain in the group (the last scan chain in the group wraps around to the first) and then including some feedbacks to implement a primitive polynomial for a modular (type II) LFSR. Each of the large horizontal LFSRs essentially has *n/b* external inputs that are fed by the corresponding *n/b* stages in the small vertical LFSR. Each of these external inputs in the large horizontal LFSR enters at a distance of *m-1* stages away from each other in the LFSR structure. An optional phase-shifter between the small vertical LFSR and the large horizontal LFSRs can also be implemented by adding some additional XOR terms [Rajski 98b].

Not counting the feedback taps for implementing the primitive polynomials or optional phase shifter, the number of XOR gates required for configuring this architecture consists of the following. The small vertical LFSR has *n* 2-input XORs, each of the large horizontal LFSRs has *n/b* 2-input XORs, and the linear combinational expansion circuit has *n* 3-input XORs. The number of XOR gates required for implementing the primitive polynomials will be *(b+1)* times *(number of terms per polynomial - 1)*. Each of the XORs in the small vertical LFSR has one of its inputs (the one creating feedback) ANDed with a control signal (*enable_small_LFSR*). The same is done for the large horizontal LFSRs with a control signal (*enable_large_LFSRs*). The *enable_small_LFSR* control signal is set to 0 to disable the feedback of the small vertical LFSR during the

first clock cycle in which the first bit-slice is loaded. This is done so that the output response from the previous scan vector does not interfere with the state of the small vertical LFSR. Essentially, the small vertical LFSR gets initialized in the first clock cycle. In subsequent clock cycles, the *enable_small_LFSR* control signal is set to 1 so that it operates as an LFSR. For the large horizontal LFSR, the *enable_large_LFSRs* control signal is set to 0 to disable its feedback during the first *m* clock cycles. Again, this is done to allow the output response to be shifted out and the large horizontal LFSRs to get initialized with linear combinations of the free variables coming in from the tester. Starting in the *m+1* clock cycle, the *enable_large_LFSR* control signal is set to 1 so that it operates as an LFSR. Note that the *enable_large_LFSR* and *enable_small_LFSR* signals are controlled by the on-chip test controller.

8.2 Operation of Proposed Decompressor

The architecture for the proposed scheme (described in Sec. 8.1) combines three different stages of linear expansion. The **first stage** is the linear combinational expansion circuit which expands the *b* free-variables that arrive from the tester each clock cycle to fill each bit-slice of the multiple scan chains. This is similar to [Bayraktaroglu 01, 03]. For test cubes where every bit-slice has sufficiently fewer than *b* specified bits, the proposed scheme can decompress the test cube in *m* clock cycles (the minimum number of cycles needed to fill the scan chains). Thus it performs very well for lightly specified test cubes.

For test cubes where one or more bit-slices have too many specified bits to solve with only *b* free-variables, the scheme in [Bayraktaroglu 01, 03] cannot be used for decompression. Thus, the scheme in [Bayraktaroglu 01, 03] either has to reduce the ratio of *n* to *b* (thereby reducing the amount of compression), or rely on special constrained ATPG to avoid such test cubes if possible. The proposed scheme avoids this problem by having a **second stage** of linear expansion through the use of the small vertical LFSR. Note that this is similar to the scheme in [Rajski 02]. The advantage of having the small vertical LFSR is that it will cycle the free-variables that are injected into it so that the set

of linear equations for each bit-slice of the scan chains will depend not just on *b* free-variables as in [Bayraktaroglu 01, 03], but on free-variables injected across multiple clock cycles. Thus the excess free variables in lightly specified bit-slices can be used to solve for the highly specified bit-slices. This adds a lot of flexibility in solving the equations thus enabling a higher encoding efficiency. It also allows the ratio of *n* to *b* to be scaled much higher without greatly reducing the space of possible test cubes that can be generated.

A minimum of *m* clock cycles are required to fill the scan chains. This will inject *mb* free-variables into the small vertical LFSR. In the best case, it may be possible to solve the system of linear equations for the test cube after *m* clock cycles. However, if the number of specified bits in the test cube is more than *mb*, then more free-variables are needed. In this case, additional clock cycles beyond what is needed to initially fill the scan chains with data are required. The key idea is that the tester keeps shifting free-variables into the LFSR until it is possible to solve the system of linear equations for the test cube. Whereas the scheme in [Rajski 02] uses a fixed number of clock cycles for generating all test cubes (whether that many are needed or not), the proposed scheme uses only as many clock cycles as are needed for each test cube. As soon as it is possible to solve the system of linear equations, then the test cube is generated and applied to the circuit-under-test (CUT). The output response can be shifted out and compacted with a multi-input signature register (MISR) as the next test cube is generated. The solution to the system of linear equations gives the values for the free-variables (i.e., the bits stored on the tester) that will produce the test cube.

At any point, the small vertical LFSR can only store as many linear combinations of free-variables as the size of the LFSR (i.e., *n*). Thus if the test cube has too many specified bits, it may not be possible to solve the system of linear equations even if it is run for an infinite number of clock cycles because of linear dependencies. To avoid putting any limits on the number of specified bits per test cube, the proposed scheme uses a **third stage** of linear expansion through the use of the large horizontal LFSRs. Each of these large horizontal LFSRs has *n/b* external inputs coming from the small vertical LFSR and indirectly from the outputs of the linear combinational expansion circuit. Let

one of the *n/b* inputs to each of the *b* large horizontal LFSRs be labeled as I_1, I_2, …, I_b, as shown in Fig. 8.2. Then in the proposed scheme, the linear combinational expansion circuit is designed in such a way that this set of *b* inputs taken together form a linearly independent set of equations in terms of the *b* free-variables arriving each clock cycle. Since there are *b* free-variables arriving each clock cycle, and *b* large horizontal LFSRs, it is always possible to do this. In the trivial case, each of the inputs, I_1, I_2, …, I_b, could be directly connected to one of the *b* inputs coming from the tester (in the general case, linearly independent equations in terms of say 3 free-variables each could be constructed). This ensures that the contents of each of the *b* large horizontal LFSRs can be loaded independently with any specified set of values. In at most *(n/b)(m)* clock cycles, the set of large horizontal LFSRs could be put into any possible state. In the worst case, it is equivalent to having a separate serial input to each of the *b* large horizontal LFSRs that serially loads each LFSR with whatever the desired state is.

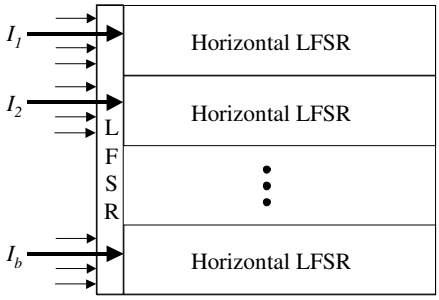


Figure 8.2. Set of *b* Linearly Independent Inputs

If it is necessary to use the proposed scheme to load fully specified test cubes, then the small vertical LFSR could be implemented separately from the scan cells such that all the scan cells are included in only the large horizontal LFSRs. In this case, any test cube (even fully specified test cubes) can be encoded with no more than *(n/b)(m)* clock cycles. Note that *(n/b)(m)* is equal to the number of clock cycles that would be required to load the scan chains with *b* channels from the tester if no compression was used. When the small vertical LFSR is constructed from the scan cells themselves as shown in Fig. 8.1, then there may be some fully or nearly fully specified test cubes that could not be encoded because of linear dependencies between the small vertical LFSR and the large

horizontal LFSRs. In general, the proposed scheme would not be used for nearly fully specified test cubes, and thus constructing the small vertical LFSR from the scan cells themselves will not cause any problem.

One question that may arise is given the fact that all the test cubes can be generated just using the large horizontal LFSRs, why bother with the small vertical LFSR. The reason is that for less specified test cubes, using only the large horizontal LFSRs may have much less encoding efficiency because it takes *m* cycles before the large horizontal LFSRs are initialized and the feedback is enabled. Thus the large horizontal LFSRs are much slower in cycling the free-variables from multiple clock cycles compared with the small vertical LFSR. One of the powerful aspects of the proposed scheme is combining all three stages of linear expansion together so that it achieves high encoding efficiency regardless of the distribution of specified bits in the test cubes.

Given a test cube with *s* specified bits, it can be assumed that at least $\lceil s/b \rceil$ clock cycles will be needed before there is any chance of solving it. The linear decompressor is symbolically simulated for $\lceil s/b \rceil$ clock cycles. The system of linear equations is then formed for the *s* specified bits and a linear equation solver is called to try to solve it. If it cannot be solved, then the linear decompressor is simulated for another clock cycle (adding more free-variables) and a new system of linear equations is formed and an attempt is made to solve it. This continues until a solution to the system of linear equations is found. Note that the symbolic simulation process, forming the system of linear equations, and solving it can all be done very quickly.

8.3 Transmitting Control Information

In the proposed scheme, since the number of clock cycles needed to decompress each test cube is variable, this information needs to be transmitted to the on-chip test controller so it knows when to apply the system clock. The on-chip test controller has a *SCAN_COUNTER* which counts how many clock cycles have elapsed since the last scan vector was applied to the CUT. There also is a *FINAL_COUNT* register that stores the number of clock cycles that it will take to generate the current test cube. This is

compared with the contents of the *SCAN_COUNTER* to determine when the test cube should be applied to the CUT. The value of the FINAL_COUNT for each test cube can be transmitted in the first clock cycle(s) when generating the test cube. The first b bits coming from the tester for each test cube can be loaded into the FINAL_COUNT register to initialize it for each test cube. Alternatively, if the test cubes are ordered so that the value of FINAL_COUNT monotonically increases between test cubes, then only the increment in FINAL_COUNT between two test cubes needs to be transmitted.

8.4 Analysis

One question to consider is given a set of test cubes, when does varying the number of free-variables per test cube pay off compared with just using a fixed number of free-variables. Using a fixed number of free-variables requires no control bits per test cube. Using a variable number of free-variables per test cube with the proposed scheme should require less than $log_2(s_{avg})$ control bits per test cube. Thus the loss in encoding efficiency due to the control bits is less than $log_2(s_{avg})/s_{avg}$. This is plotted with respect to s_{avg} in Fig. 8.3. As can be seen, the loss in encoding efficiency due to the control bits diminishes rapidly as s_{avg} increases. For a circuit with 100,000 scan cells and only 1% specified bits, s_{avg} is 1,000 which corresponds to less than 1% loss in encoding efficiency. As either the number of scan cells goes up, or the percentage of specified bits goes up, the loss in encoding efficiency becomes even less. Thus the control bits will have a fairly negligible effect on the amount of compression for industrial circuits.

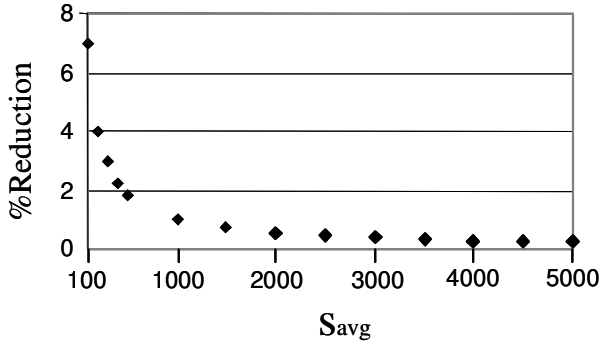


Figure 8.3. Percentage Reduction in Encoding Efficiency Due to Control Bits for Different Values of s_{avg}

Ignoring the impact of the control bits, then the improvement in compression that is achieved using a variable number of free-variables per test cube versus a fixed number will be approximately s_{max}/s_{avg} assuming an encoding efficiency of approximately 1. Thus, for a set of test cubes that are about 2% specified (s_{avg} = 2%) with the worst-case test cube being 4% specified (s_{max} = 4%), the amount of compression would be roughly two times greater. If s_{max} differs significantly from s_{avg}, then varying the number of free-variables per test cube can provide a substantial improvement in compression.

If the ATPG and compaction procedures can be constrained to keep s_{avg} and s_{max} close, then using a fixed number of free-variables per test cube retains the advantage of less control complexity. However, the proposed scheme is attractive for situations where constraining the ATPG is either not possible, not desirable, or not efficient. For some circuits, allowing unconstrained ATPG with no limit on static and dynamic compaction can significantly reduce the total number of test cubes and hence the total number of specified bits in the test set resulting in significantly better compression using the proposed scheme. For some circuits, some faults may require a substantial number of specified bits in order to detect, thus making it difficult to keep s_{avg} and s_{max} close. Finally, in some scenarios, it may be desirable to re-use a set of test cubes that was not generated with constrained ATPG.

8.5 Experimental Results

To see how the encoding efficiency for the proposed 3-stage scheme varies with respect to different distributions of specified bits and different scan architectures, the first set of experiments was performed on randomly generated test cubes. The results are shown in Table 8.1. The number of channels from the tester, b, was 16. Keeping the total number of scan cells constant across the different architectures, the number of scan chains, n, was taken as 64, 128, 256, 512, and 1024. Randomly generated test cubes were encoded and the encoding efficiency was calculated as the number of specified bits divided by the number of bits required to encode them. Results are shown for both the 3-stage scheme as well as a 2-stage decompressor (only combinational expander and

vertical LFSR) both using a variable number of free-variables per test cube. The reason for showing both the 2-stage and 3-stage case is to illustrate the importance of the 3rd stage in achieving high encoding efficiency for non-uniform specified bit distributions where the number of specified bits per test cube varies. The first row for each scheme corresponds to the case where the number of specified bits per test cube varied randomly in the range between 2-5%. The second row is for 2-10%, followed by 2-20%, and 2-50%.

While the 3-stage scheme is able to fully encode all the test cubes regardless of the scan architecture and the specified bit distribution, the two stage scheme is unable to perform similarly ('--' indicates the cases for which encoding was not possible). Note that as the number of scan chains increases, the encoding efficiency for the 3-stage scheme increases significantly. This also has the advantage of reducing the test time, since the length of each scan chain is smaller for the architectures having larger number of scan chains. For the two stage scheme to be able to encode all the test cubes, the number of scan chains needs to be small enough so that the most specified test cubes can be encoded. However, having a smaller number of scan chains reduces the encoding efficiency for the less specified test cubes. For example, for test cubes with 5% specified bits, the 2-stage scheme cannot encode them with 512 scan chains, so the number of scan chains has to be reduced to 256. However, this limits the encoding efficiency that can be achieved for the 2% specified test cubes in the same test set. Thus, the overall encoding efficiency for a range of test cubes between 2%-5% is limited to 0.56. The 3-stage scheme also has similarly limited encoding efficiency with 256 scan chains, however, the advantage of the 3-stage scheme is that the number of scan chains can be increased to 1024 while still being able to solve for the most specified test cubes. By using 1024 scan chains, an encoding efficiency of 0.99 can be achieved even for 2% specified test cubes. Thus with the proposed 3-stage scheme, the encoding efficiency can consistently be made high regardless of the distribution of specified bits by using a scan architecture with a sufficiently large number of scan chains. Note that if the number of specified bits is kept relatively uniform across all test cubes by constraining the ATPG (as is done in the methodology proposed in [Rajski 02]), then a 2-stage scheme works very well and there

is no need to go to a 3-stage scheme. The advantage of the 3-stage scheme comes only when the specified bit distribution is non-uniform.

Table 8.1. Comparison between Two-Stage and Three-Stage Linear Decompressors for Non-Uniform Specified Bit Distributions

Linear Decompressor	Random Test Cubes	Scan Architecture				
	Specified Bit Distribution	n=64 m=2048	n=128 m=1024	n=256 m=512	n=512 m=256	n=1024 m=128
2-Stage	2%-5%	.14	.28	**.56**	--	--
	2%-10%	.24	**.48**	--	--	--
	2%-20%	**.44**	--	--	--	--
	2%-50%	--	--	--	--	--
3-Stage	2%-5%	.14	.28	.56	.90	**.99**
	2%-10%	.24	.48	.80	.95	**.99**
	2%-20%	.44	.74	.90	.97	**.99**
	2%-50%	.77	.90	.96	**.99**	**.99**

A second set of experiments were performed on the largest ISCAS 89 benchmark circuits. For each circuit, ATPG was performed to generate test cubes for the non-redundant faults. The set of test cubes was encoded using the proposed scheme assuming 8 channels coming from the tester and expanding to fill the scan chains. The results are shown in Table 8.2. The total number of specified bits in each test vector is shown followed by the number of bits that need to be stored on the tester for the proposed scheme. Note that the proposed method has a high encoding efficiency bringing the test data storage requirements down close to the number of specified bits. The encoding efficiency is not as high as in Table 8.1 because for these small circuits, s_{avg} is small such that the control bits have a more significant impact on the encoding efficiency.

A comparison was made with the scheme in [Bayraktaroglu 03] in Table 8.3. This scheme uses only a linear combinational circuit. It requires a special constrained ATPG to ensure all test cubes can be encoded. As can be seen, the proposed scheme significantly reduces the tester storage requirements by configuring the scan cells into an LFSR.

Table 8.4 shows a comparison of the results for the proposed scheme with a variety of other test data compression schemes. The proposed scheme requires less test data storage

for all circuits in comparison with these approaches. Note that it was not possible to directly compare results for the proposed scheme with those in [Könemann 01], [Rajski 02], and [Volkerink 03], since those papers only show results for circuits that are not publicly available.

Table 8.2. Results for Proposed Scheme Using 8 Tester Channels

Circuit Name	Num Scan Chains	Num. Test Cubes	Total Num. Bits	Num. Specified Bits	Test Storage (bits)	Encoding Efficiency
s13207	175	255	178,500	9,335	11,320	0.82
s15850	153	142	86,762	10,452	11,584	0.90
s38417	185	105	174,720	29,847	30,560	0.98
s38584	183	192	281,088	25,636	27,248	0.94

Table 8.3. Comparison with [Bayraktaroglu 03]

Circuit		[Bayraktaroglu 03]			Proposed Scheme		
Name	Scan Chains	Num. Channels	Num. Vect.	Test Storage	Num. Channels	Num. Vect.	Test Storage
s13207	175	19	258	19,608	8	255	11,320
s15850	153	18	167	12,024	8	142	11,584
s38417	185	19	317	54,207	8	105	30,560
s38584	183	19	185	28,120	8	192	27,248

Table 8.4. Comparison of Test Data for Different Encoding Schemes

Circuit Name	MinTest [Hamzaoglu 98]		Illinois Scan Architecture [Hamzaoglu 99]		FDR Codes [Chandra 01]		Mutation Encoding [Reda 02]		Seed Overlapping [Rao 03]		Proposed Scheme	
	Num. Vect.	Total Bits	Num. Vect.	Total Bits	Num. Vect.	Total Bits	Num. Vect.	Total Storage	Num. Vect.	Total Bits	Num. Vect.	Total Bits
s13207	233	163,100	273	109,772	236	30,880	274	16,913	272	17,970	255	11,320
s15850	96	58,656	178	32,758	126	26,000	185	14,676	174	15,774	142	11,584
s38417	68	113,152	337	96,269	99	93,466	231	55,848	288	60,684	105	30,560
s38584	110	161,040	239	96,056	136	77,812	220	47,886	215	31,061	192	27,248

Chapter 9

Summary and Future Work

The current trend in the semiconductor industry is to move towards very deep sub-micron technologies to increase functionality and achieve higher performance. Thus the amount of data required to test these large designs has increased tremendously. Unfortunately, the automatic test equipment has not been able to keep up with the large tester storage and bandwidth that is required for such designs. Hence conventional external testing is becoming increasingly difficult. One way to overcome this problem is to use pseudo-random built-in self-test (BIST). While pseudo-random BIST can significantly reduce the tester storage requirements, its drawbacks include long test times and insufficient fault coverage. Ways to tackle this involve modification to functional logic and/or large hardware overhead, both of which are significant drawbacks of stand-alone BIST. Another approach is to perform mixed-mode testing where pseudo-random BIST can be used to detect the easy (random-pattern-testable) faults, and deterministic vectors targeting only the hard faults are stored on the tester. But it has been seen that the storage for even these "top-up" vectors on the tester is as much as 30%-50% of the storage required for external testing.

Linear expansion techniques provide an attractive alternative since they are able to significantly reduce the tester storage requirements while requiring very little hardware overhead on the chip. These techniques also help to reduce the tester bandwidth requirements since they require much fewer channels from the tester as compared to external testing. One of the restrictions of external testing is that the number of channels from the tester limits the number of scan chains within the circuit. Since linear expansion techniques can expand the data from a few tester channels to a large number of scan chains, the number of scan chains can be increased, which reduces the length of each scan chain, and hence reducing the test time required to load a test vector. These techniques also do not require modification of the functional logic, which is something very desirable from a designer's point of view.

Chapter 2 classified the existing linear expansion schemes based on two attributes: whether they obtain data every clock cycle (continuous-flow) or only for a portion of the clock cycles required to load the scan chains (periodically-loaded), and whether the data stored on the tester for every test cube is the same or variable.

Continuous-flow schemes can be very efficiently connected to the tester and are also nice from a tools integration standpoint since they mimic the standard behavior of normal scan chains. Periodically-loaded schemes typically can achieve slightly higher encoding efficiency, but they require some test scheduling to fully utilize the tester bandwidth during the time period between loads. They are thus well suited for a system-on-chip (SOC) environment where multiple cores are being tested concurrently. The schemes described in chapters 4, 5, and 6 were periodically-loaded schemes whereas chapters 7 and 8 described two continuous-flow schemes.

Since most periodically-loaded schemes are based on reseeding of LFSRs, chapter 3 described the idea of LFSR reseeding in greater detail. Previous forms of LFSR reseeding have been static and have required full reseeding. Chapter 4 showed how partial dynamic LFSR reseeding provides much higher encoding efficiencies than previous forms of LFSR reseeding while requiring low hardware overhead. The linear equations were solved in a manner such that the effect of the variance in specified bits per test cube is minimized. Even though the scheme gives very good results, using a special ATPG procedure could improve them further. Another area for improvement would be to reduce the computation time by generating and solving the linear equations in a more optimal manner.

Chapter 5 then showed how a generalized form of partial reseeding allows the designer to tradeoff between tester storage and test application time based on the available test resources. Unlike the scheme in [Dorsch 01b], this scheme avoids the need for a special ATPG procedure by utilizing the degrees of freedom for embedding the test cubes within the pseudo-random sequence.

Chapter 6 described how LFSR reseeding and statistical coding can be combined in a clever way. This work showed how the large solution space of linear equations can be exploited to find LFSR seeds that can be efficiently encoded using statistical codes. This

scheme can be used for different applications by implementing it with the different architectures that were described. One area for future research would be to investigate other codes that can be used to efficiently compress the LFSR seeds.

One of the drawbacks of combinational linear expansion networks is that the data coming from the tester every clock cycle should be enough to encode the most specified bit-slice of the scan chains. Since this leads to an inefficient utilization of the test data, the work described in chapter 7 showed how the capability to adjust the width of linear expansion every clock cycle allows us to eliminate this requirement, and achieve greater compression than fixed width expansion techniques. This scheme also has the nice feature that it can generate any test cube, regardless of the number of specified bits in it. Future research could look at improving the compression by using a special ATPG procedure, and by trying to encode multiple bit-slices within a single tester clock cycle.

Chapter 8 then showed how three stages of linear expansion can be combined to effectively compress any test cube regardless of the distribution of specified bits. No special constraints on the ATPG process were necessary. The hardware requirements for this scheme are small since it makes use of the scan cells themselves to configure the decompressor.

The end result of the work described in this dissertation is a set of new test data compression techniques based on linear expansion that are able to significantly reduce the tester storage and bandwidth requirements. Linear expansion schemes can be simple to implement, yet very effective, as seen from the work described in this dissertation. Some of the schemes can encode any test cube regardless of the number of specified bits, which is very useful in case there are any last minute design changes and the test cubes have to be regenerated. Another nice feature of these schemes is that they do not require any special ATPG procedure to generate the test cubes, which greatly simplifies matters from a tool integration standpoint. If a special ATPG process is used, then the results obtained using these schemes could be improved further.

As more and more chips are manufactured using nanometer technology, delay-inducing defects are causing increasing concern in the semiconductor industry. Studies have shown that an increasingly large amount of test data volume and test application

time is required for testing these chips using the transition-fault delay model. Hence future research could investigate newer techniques for compressing vectors that detect delay faults. Another primary concern is the amount of power dissipation during the testing process. In order to minimize power, it is desirable to reduce the amount of switching activity that takes place within the circuit while the test vectors are being shifted into the scan chains. Future research could study the impact of linear expansion techniques on power dissipation, and come up with decompressors that jointly minimize power and test storage. Since the techniques described in this dissertation were for scan based testing, further research can be done in developing techniques that can efficiently compress functional vectors. While there is ongoing research in the above mentioned areas, more advanced techniques are needed to handle the growing size and complexity of the circuits to be tested.

.

Bibliography

[Bardell 82] Bardell, P.H., and W.H. McAnney "Self-Testing of Multichip Logic Modules," *Proc. of International Test Conference*, pp. 200-204, 1982.

[Bardell 87] Bardell, P.H., and W.H. McAnney, and J. Savir, *Built-in Test for VLSI: Pseudorandom Techniques*, John Wiley & Sons, 1987.

[Bayraktaroglu 01] Bayraktaroglu, I., and A. Ogailoglu, "Test Volume and Application Time Reduction Through Scan Chain Concealment," *Proc. of Design Automation Conference*, pp. 151-155, 2001.

[Bayraktaroglu 03] Bayraktaroglu, I., and A. Ogailoglu, "Decompression Hardware Determination for Test Volume and Time Reduction through Unified Test Pattern Compaction and Compression," *Proc. of VLSI Test Symposium*, pp. 113-118, 2003.

[Bershteyn 93] Bershteyn, M., "Calculation of Multiple Sets of Weights for Weighted Random Testing," *Proc. of International Test Conference*, pp. 1031-1040, 1993.

[Brglez 89] Brglez, F., D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," *Proc. of International Symposium on Circuits and Systems*, pp. 1929-1934, 1989.

[Chandra 00] Chandra, A., and K. Chakrabarty, "Test Data Compression for System-on-a-Chip Using Golomb Codes," *Proc. of VLSI Test Symposium*, pp. 113-120, 2000.

[Chandra 01] Chandra, A., and K. Chakrabarty, "Frequency-Directed Run Length (FDR) Codes with Application to System-on-a-Chip Test Data Compression," *Proc. of VLSI Test Symposium*, pp. 42-47, 2001.

[Chatterjee 95] Chatterjee, M., and D.K. Pradhan, "A Novel Pattern Generator for Near-Perfect Fault Coverage," *Proc. of VLSI Test Symposium*, pp. 417-425, 1995.

[Chen 86] Chen, C.L., "Linear Dependencies in Linear Feedback Shift Registers", *IEEE Transactions on Computers*, Vol. C-35, No. 12, pp. 1086-1088, Dec. 1986.

[Cullen 97] Cullen, C.G., *Linear Algebra with Applications*, Addison-Wesley, ISBN 0-673-99386-8, 1997.

[Das 00] Das, D., and N.A. Touba, "Reducing Test Data Volume Using External/LBIST Hybrid Test Patterns," *Proc. of International Test Conference*, pp. 115-122, 2000.

[Dorsch 01a] Dorsch, R., and H.-J. Wunderlich, "Reusing Scan Chains for Test Pattern Decompression," *Proc. of European Test Workshop (ETW)*, May 2001.

[Dorsch 01b] Dorsch, R., and H.-J. Wunderlich, "Tailoring ATPG for Embedded Testing," *Proc. of International Test Conference*, pp. 530-537, 2001.

[Eichelberger 83] Eichelberger, E.B., and E. Lindbloom, "Random-Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Test," *IBM Journal of Research & Development*, Vol. 27, No. 3, pp. 265-272, May 1983.

[Fagot 98] Fagot, C., P. Girard, and C. Landrault, "On Using Machine Learning for Logic BIST," *Proc. of International Test Conference*, pp. 338-346, 1998.

[Gonciari 02] Gonciari, P.T., B. Al-Hashimi, and N. Nicolici, "Improving Compression Ratio, Area Overhead, and Test Application Time for System-on-a-Chip Test Data Compression/Decompression," *Proc. of Design Automation and Test in Europe (DATE)*, pp. 604-611, 2002.

[Hamzaoglu 98] Hamzaoglu, I., and J. Patel, "Test Set Compaction Algorithms for Combinational Circuits," *Proc. of International Conference on Computer-Aided Design (ICCAD)*, pp. 283-289, 1998.

[Hamzaoglu 99] Hamzaoglu, I., and J.H. Patel, "Reducing Test Application Time for Full Scan Embedded Cores," *Proc. of Int. Symposium on Fault Tolerant Computing*, pp. 260-267, 1999.

[Hellebrand 92] Hellebrand, S., S. Tarnick, J. Rajski, and B. Courtois, "Generation of Vector Patterns Through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers," *Proc. of International Test Conference*, pp. 120-129, 1992.

[Hellebrand 95a] Hellebrand, S., J. Rajski, S. Tarnick, S. Venkataraman and B. Courtois, "Built-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers," *IEEE Transactions on Computers*, Vol. 44, No. 2, pp. 223-233, Feb. 1995.

[Hellebrand 95b] Hellebrand, S., B. Reeb, S. Tarnick, and H.-J. Wunderlich, "Pattern Generation for a Deterministic BIST Scheme," *Proc. of International Conference on Computer-Aided Design (ICCAD)*, pp. 88-94, 1995.

[Hellebrand 00] Hellebrand, S., H.-G. Liang, and H.-J. Wunderlich, "A Mixed Mode BIST Scheme Based on Reseeding of Folding Counters," *Proc. of International Test Conference*, pp. 778-784, 2000.

[Hetherington 99] Hetherington, G., T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan and J. Rajski, "Logic BIST for Large Industrial Designs: Real Issues and Case Studies," *Proc. of International Test Conference*, pp. 358-367, 1999.

[Hsu 01] Hsu, F.F., K. M. Butler, J. H. Patel, "A Case Study on the Implementation of the Illinois Scan Architecture," *Proc. of International Test Conference*, pp. 538-547, 2001.

[Huffman 52] Huffman, D.A., "A Method for the Construction of Minimum Redundancy Codes," *Proc. of IRE*, Vol. 40, No. 9, pp. 1098-1101, Sep. 1952.

[Jas 98] Jas, A., and N.A. Touba, "Test Vector Decompression Via Cyclical Scan Chains and Its Application to Testing Core-Based Designs", *Proc. of IEEE International Test Conference*, pp. 458-464, 1998.

[Jas 99] Jas, A., J. Ghosh-Dastidar, and N. A. Touba, "Scan Vector Compression/Decompression Using Statistical Coding," *Proc. Of VLSI Test Symposium,* pp. 114-120, 1999.

[Jas 00] Jas, A., B. Pouya, and N.A. Touba, "Virtual Scan Chains: A Means for Reducing Scan Length in Cores", Proc. of VLSI Test Symposium, pp. 73-78, 2000.

[Jas 01] Jas, A., C.V. Krishna, and N.A. Touba, "Hybrid BIST Based on Weighted Pseudo-Random Testing: A New Test Resource Partitioning Scheme," *Proc. of VLSI Test Symposium,* pp. 2-8, 2001

[Khoche 00] Khoche, A., and J. Rivoir, "I/O Bandwidth Bottleneck for Test: Is it Real?," *Proc. of International Workshop on Test Resource Partitioning*, 2000.

[Khoche 02] Khoche, A., E.H. Volkerink, J. Rivoir, and S. Mitra, "Test Vector Compression Using EDA-ATE Synergies," *Proc. of VLSI Test Symposium*, pp. 97-102, 2002.

[Kiefer 98] Kiefer, G., and H.-J. Wunderlich, "Deterministic BIST with Multiple Scan Chains," *Proc. of International Test Conference*, pp. 1057-1064, 1998.

[Kiefer 00] Kiefer, G., H. Vranken, E.J. Marinissen, and H.-J. Wunderlich, "Application of Deterministic Logic BIST on Industrial Circuits," *Proc. of International Test Conference*, pp. 105-114, 2000.

[Könemann 91] Könemann, B., "LFSR-Coded Test Patterns for Scan Designs," *Proc. of European Test Conference*, pp. 237-242, 1991.

[Könemann 00] Könemann, B., "Logic DFT and Test Resource Partitioning for 100M Gate ASICs," *Proc. of International Workshop on Test Resource Partitioning*, 2000.

[Könemann 01] Könemann, B., "A SmartBIST Variant with Guaranteed Encoding" *Proc. of Asian Test Symposium*, pp. 325-330, 2001.

[Krasniewski 89] Krasniewski, A., and S. Pilarski, "Circular Self-Test Path: A Low-Cost BIST Technique for VLSI Circuits," *IEEE Trans. on Computer-Aided Design*, Vol. 8, No. 1, pp. 46-55, Jan. 1989.

[Krishna 01] Krishna, C.V., A. Jas, and N.A. Touba, "Test Vector Encoding Using Partial LFSR Reseeding", *Proc. of IEEE International Test Conference*, pp. 885-893, 2001.

[Krishna 02] Krishna, C.V., and N.A. Touba, "Reducing Test Data Volume Using LFSR Reseeding with Seed Compression", To appear in *Proc. of IEEE International Test Conference*, 2002.

[Krishna 03a] Krishna, C.V., and N.A. Touba, "Hybrid BIST Using an Incrementally Guided LFSR", *Proc. of IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 217-224, 2003.

[Krishna 03b] Krishna, C.V., and N.A. Touba, "Adjustable Width Linear Combinational Scan Vector Decompression", *Proc. of International Conference on Computer-Aided Design*, pp. 863-866, 2003.

[Krishna 04] Krishna, C.V., and N.A. Touba, "3-Stage Variable Length Continuous-Flow Scan Vector Decompression Scheme", to appear in *Proc. of VLSI Test Symposium*, 2004.

[Li 03] Li, L., and K. Chakrabarty, "Test Data Compression Using Dictionaries with Fixed-Length Indices," *Proc. of VLSI Test Symposium*, pp. 219-224, 2003.

[Liang 01] Liang, H.-G., S. Hellebrand, and H.-J. Wunderlich, "Two-Dimensional Test Data Compression for Scan-Based Deterministic BIST," *Proc. of International Test Conference*, pp. 894-902, 2001.

[Mitra 03] Mitra, S., and K.S. Kim, "XMAX: X-tolerant architectures for Maximal Test Compression," *Proc. of International Conference on Computer Design*, pp. 326-330, 2003.

[Mrugalski 03] Mrugalski, G., J. Rajski, and J. Tyszer, "High Speed Ring Generators and Compactors of Test Data," *Proc. of VLSI Test Symposium*, pp. 57-62, 2003.

[Muradali 90] Muradali, F., V.K. Agarwal, and B. Nadeau-Dostie, "A New Procedure for Weighted Random Built-In Self-Test," *Proc. of International Test Conference*, pp. 660-668, 1990.

[Pomeranz 93] Pomeranz, I., and S.M. Reddy, "3-Weight Pseudo-Random Test Generation Based on a Deterministic Test Set for Combinational and Sequential Circuits," *IEEE Transactions on Computer-Aided Design*, Vol. 12, No. 7, pp. 1050-1058, Jul. 1993.

[Pressly 99] Pressly, M., D. Das, and C. Hunter, "LBIST for PowerPC™ Embedded Core Microprocessors: Feasible or Not?," *International Workshop on Microprocessor Test and Verification*, 1999.

[Rajski 98a] Rajski, J., J. Tyszer, and N. Zacharia, "Test Data Decompression for Multiple Scan Designs with Boundary Scan", *IEEE Transactions on Computers*, Vol. 47, No. 11, pp. 1188-1200, Nov. 1998.

[Rajski 98b] Rajski, J., N. Tamarapalli, and J. Tyszer, "Automated Synthesis of Large Phase Shifters for Built-In Self-Test", *Proc. of International Test Conference*, pp. 1047-1056, 1998.

[Rajski 99] Rajski, J., and J. Tyszer, "A Parallel Decompressor and Related Method and Apparatuses", USA Patent #5,991,909, 1999.

[Rajski 02] Rajski, J., *et al.,* "Embedded Deterministic Test for Low Cost Manufacturing Test," *Proc. of Int. Test Conf.*, pp. 301-310, 2002.

[Rao 03] Rao, W., I. Bayraktaroglu, and A. Orailoglu, "Test Application Time and Volume Compression through Seed Overlapping," *Proc. of Design Automation Conference*, pp. 732-737, 2003.

[Reda 02] Reda, S., and A. Orailoglu, "Reducing Test Application Time Through Test Data Mutation Encoding", *Proc. of Design, Automation, and Test in Europe*, pp. 387-393, 2002.

[Reddy 02] Reddy, S., K. Miyase, S. Kajihara, and I. Pomeranz, "On Test Data Volume Reduction for Multiple Scan Chain Designs", *Proc. of VLSI Test Symposium*, pp. 103-108, 2002.

[Stroud 88] Stroud, C.E., "Automated BIST for Sequential Logic Synthesis," *IEEE Design & Test of Computers*, pp. 22-32, Dec. 1988.

[Touba 95a] Touba, N.A., and E.J. McCluskey, "Transformed Pseudo-Random Patterns for BIST," *Proc. of VLSI Test Symposium*, pp. 410-416, 1995.

[Touba 95b] Touba, N.A., and E.J. McCluskey, "Synthesis of Mapping Logic for Generating Transformed Pseudo-Random Patterns for BIST," *Proc. of International Test Conference*, pp. 674-682, 1995.

[Touba 96a] Touba, N.A., and E.J. McCluskey, "Test Point Insertion Based on Path Tracing," *Proc. of VLSI Test Symposium*, pp. 2-8, 1996.

[Touba 96b] Touba, N.A., and E.J. McCluskey, "Altering a Pseudo-Random Sequence of Bits for Scan-Based BIST", *Proc. of International Test Conference*, pp. 167-175, 1996.

[Venkataraman 93] Venkataraman, S., J. Rajski, S. Hellebrand, and S. Tarnick, "An Efficient BIST Scheme Based on Reseeding of Multiple Polynomial Linear Feedback Shift Registers," *Proc. of International Conference on Computer-Aided Design (ICCAD)*, pp. 572-577, 1993.

[Volkerink 02] Volkerink, E.H., A. Khoche, and S. Mitra, "Packet-based Input Test Data Compression Techniques," *Proc. of International Test Conference*, pp. 154-163, 2002.

[Volkerink 03] Volkerink, E.H., and S. Mitra, "Efficient Seed Utilization for Reseeding Based Compression," *Proc. of VLSI Test Symposium*, pp. 232-237, 2003.

[Wolff 02] Wolff, F.G., and C. Papachristou, "Multiscan-based Test Compression and Hardware Decompression Using LZ77," *Proc. of International Test Conference*, pp. 331-339, 2002.

[Zacharia 95] Zacharia, N., J. Rajski, and J. Tyszer, "Decompression of Test Data Using Variable-Length Seed LFSRs," *Proc. of VLSI Test Symposium*, pp. 426-433, 1995.

[Zacharia 96] Zacharia, N., J. Rajski, J. Tyszer, and J. Waicukauski "Two Dimensional Test Data Decompressor for Multiple Scan Designs," *Proc. of International Test Conference*, pp. 186-194, 1996.

Vita

Krishna Vijaya Chakravadhanula was born in Razole, India on November 25, 1976, the son of Lakshmi Annapurna Chakravadhanula and Murthy Chakravadhanula. After completing high school at St. Thomas' Boys School, Calcutta, India, in 1993, and senior secondary also at St. Thomas' Boys School, Calcutta, India, in 1995, he entered the College of Engineering, Jadavpur University, Calcutta. He majored in Computer Science and Engineering and got the Bachelor of Engineering (B.E.) degree from Jadavpur University in June 1999. In August 1999, he entered the Graduate Program at the University of Texas at Austin, Texas to pursue an M.S./Ph.D. in Electrical and Computer Engineering. He received an M.S. degree in Electrical and Computer Engineering from the University of Texas at Austin in May 2001. During his graduate studies, he was a recipient of the IBM Ph.D. Fellowship Award for the academic years 2002-2003 and 2003-2004. He also worked as a summer intern at Synopsys, Agere Systems and IBM.

Permanent Address: 2-18-20 Madhav Nagar,
Kakinada – 533003,
Andhra Pradesh, India.

This dissertation was typed by the author.