A Reconfigurable Shared Scan-In Architecture

by

Samitha Samaranayake

S.B. Computer Science, Massachusetts Institute of Technology (2002)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2003

Author ______________________________
Department of Electrical Engineering and Computer Science
May 22, 2003

Certified by ______________________________
Rohit Kapur
Principal Engineer, Test Research and Development, Synopsys Inc.
Thesis Supervisor

Certified by ______________________________
Volker Strumpen
Research Scientist, MIT Laboratory for Computer Science
Thesis Supervisor

Accepted by ______________________________
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

A Reconfigurable Shared Scan-In Architecture

by

Samitha Samaranayake

Submitted to the Department of Electrical Engineering and Computer Science
on May 30, 2003, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis presents a test architecture for reducing the cost of testing an integrated circuit (IC). The cost of testing an IC is based on the volume of test data that needs to be loaded during the testing process and the time required to test the circuit. The proposed solution reduces both these factors using a new architecture to load test data in parallel. Typically, test data is loaded into a circuit using chains of internal storage elements, called scan chains and each of these chains is connected to the inputs of the circuit under test. When loading the test data in parallel, many scan chains share the same input. The scan chains that share the same input must contain the same data, creating constraints on the test data that can be loaded. The constraints introduced increase as the number of chains that are loaded in parallel increases. Therefore, certain defects in a circuit might not be testable, if a large number of chains are loaded in parallel. The proposed architecture reduces the constraints of parallel loading by changing the inputs that drive each scan chain during the testing process. We call this change, reconfiguration of the scan chains. This thesis presents two methods of determining how to reconfigure the scan chains. The first method provides the largest gains, but requires a large amount of additional computation when creating the test data. Therefore, a second method that is less effective, but requires no additional computation is also presented. Experimental results show as much as a factor of 170 reductions in both the test time and test data volume relative to the standard serial testing.

Thesis Supervisors:

Rohit Kapur
Principal Engineer, Test Research and Development, Synopsys Inc.

Volker Strumpen
Research Scientist, MIT Laboratory for Computer Science

Acknowledgements

I take this opportunity to thank everyone at MIT and Synopsys who helped me by providing guidance and support during the process of creating this thesis. My MIT thesis advisor Dr. Volker Strumpen, provided me with valuable guidance. I was able to greatly improve the quality of my thesis write-up due to his constructive criticism. I thank him for his advice and patience in doing so.

The research that is the foundation of this thesis would not have been possible if not for many people at the Test Automation Products Research Department at Synopsys Inc., my VI-A company. My thesis advisor at Synopsys, Rohit Kapur, was the driving force behind this research. His insight, encouragement and positive attitude made it both interesting and enjoyable to work with him. I would like to thank Jim Sproch, the Director of Test Automation Products R&D for all his support and advice throughout the years, and for giving me the opportunity to work at Synopsys. Tom Williams was one of the strongest supporters of this research project and was always full of encouragement. My Manager Wolfgang Meyer and the other Managers Tony Taylor, Surya Duggirala and Girish Patankar were also very supportive and quick to help when I had questions in their areas of expertise. Emil Gizdarski and Frederic Neuveux were on the project team and are largely responsible for the success of this research. We would not have been able to move forward with this research if not for Emil's work on modifying the Synopsys ATPG engine TetraMAX®* to work with this architecture. Similarly, Frederic took care

* Synopsys is a registered trademark of Synopsys, Inc., TetraMAX is a registered trademark, of Synopsys, and DFT Compiler is a trademark.

of all the changes that needed to be done in the DFT Compiler™. I thank Minesh Amin for always taking time to help me, and for the interesting and entertaining discussions during breaks. I would also like to thank Michelle Beck for helping me with whatever administrative problems that I had. Last but not least, I thank Nodari Sitchinava, my fellow VI-A intern at Synopsys, for all his contributions that made this research successful.

The VI-A program at MIT provided me with an opportunity to do my thesis research at Synopsys. I thank Professor Markus Zahn and Kathy Sullivan of the VI-A program and Professor Srinivas Devadas the Synopsys VI-A advisor. I also thank Anne Hunter and Vera Sayzew of the EECS Department for all their assistance.

Finally, I would like to thank my parents for all their support, encouragement and the sacrifices they have made for me throughout the years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Advances in semiconductor technology have made Integrated Circuit (IC) components smaller and cheaper, making it possible to create larger and more complex designs. Figure 1-1 shows that the per transistor cost of manufacturing has been falling steadily, while the per transistor cost of testing IC's has remained relatively unchanged [1]. As a result, the cost of testing a chip is increasing relative to the total cost of design and manufacturing. It has been predicted that testing a chip will cost more than manufacturing it by 2014, unless the search for low cost test solutions changes this trend [1].

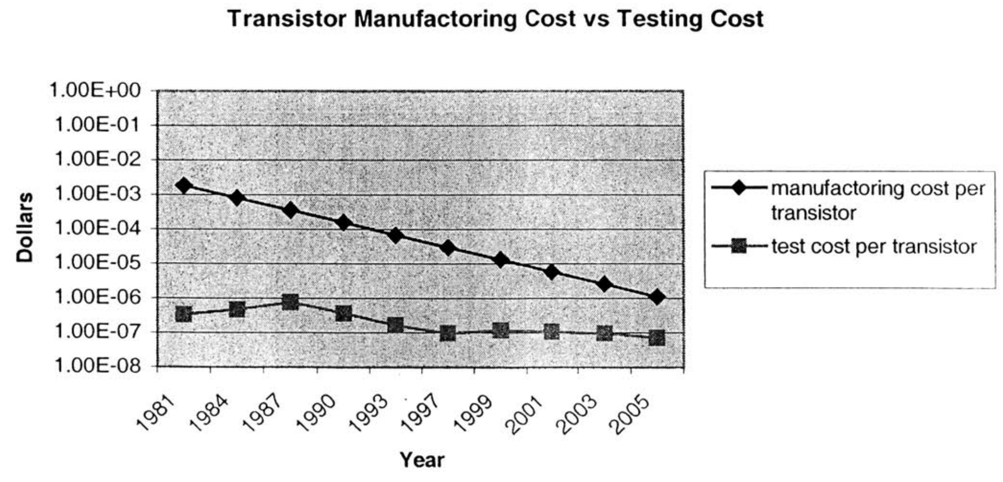


Figure 1-1*: The cost of manufacturing an IC is decreasing faster than the cost of testing a circuit [1].*

With the cost of manufacturing transistors decreasing [1], designers are now willing to add more circuitry to decrease the testing cost [2]. The ability to add such circuitry, known as Design for Test (DFT) circuitry, makes it possible to use advanced test architectures within a chip, making it easier and faster to test the chip. As the circuits being designed get larger[1], testing them requires a large volume of test data. Therefore, testing these large circuits requires very expensive[2] test equipment that can handle a large volume of data [1]. Furthermore, as the volume of test data increases the test application time increases as well, and multiple testers might be needed to test the chips in a timely manner. Therefore, DFT solutions that can reduce both test data volume and test application time, are critical for testing large circuits.

[1] Current designs can have more than 20 million gates [1].
[2] One set of testing equipment can cost over twenty million dollars [1].

1.2 Testing Integrated Circuits

This thesis focuses on the structural test of a VLSI circuit. Structural tests check whether the logic structure of the chip is fabricated as designed [2]. The purpose of such testing is to determine whether defects have been introduced during the manufacturing process. At this stage, it is assumed that the design of the chip behaves according to its specification[3].

The basic principle of structural testing is very simple. A chip is placed on a piece of Automatic Test Equipment (ATE) that can apply inputs to the circuit and observe its response. A set of binary vectors, called *test vectors*, is then applied to the circuit using the ATE [2]. If the response observed by the ATE matches the expected output of the test vector, as determined by the logic of the circuit, then that vector is said to have passed. If the response does not match the expectation, the vector is said to have failed. The chip is considered to be good only if all the test vectors pass. Even a single failed test identifies a defective chip. The confidence level that we have, as to whether a chip that passes this testing is indeed not defective, depends on the effectiveness of the testing. Testing the chip for all possible input sequences is one way of creating a test with high confidence. However, this is not feasible in general due to the large number of test vectors required. In practice, a chip is tested using a much smaller number of test vectors. These vectors might not detect all possible defects of the circuit, but try to maximize the number of possible defects that are tested.

[3] Verification, a process done prior to the fabrication of a chip, tests whether the circuit behaves according to its design specification.

A fault model is used to model the possible defects, and to create tests that maximize the detection of the modeled faults [2]. The most commonly used fault model is the *single stuck-at fault model*, which in practice, captures over 95% of all possible manufacturing defects in an IC [3]. In this model, the circuit is considered to be a collection of interconnected gates as shown in Figure 1-2. There are two possible faults for each connection, stuck-at-0 and stuck-at-1 [4]. If the connection is stuck-at-0, then the value on that wire will have a logic value of 0 regardless of the value being driven through it. Similarly, if the connection is stuck-at-1 the wire will have a logic value of 1. A circuit with n connections will have 2n potential single stuck-at faults. The faults are modeled by creating a list of all possible stuck-at faults. This list is called the *fault list* of the circuit [4]. After the fault list has been created, an Automatic Test Pattern Generation (ATPG) engine is used to create test vectors that can test these faults.

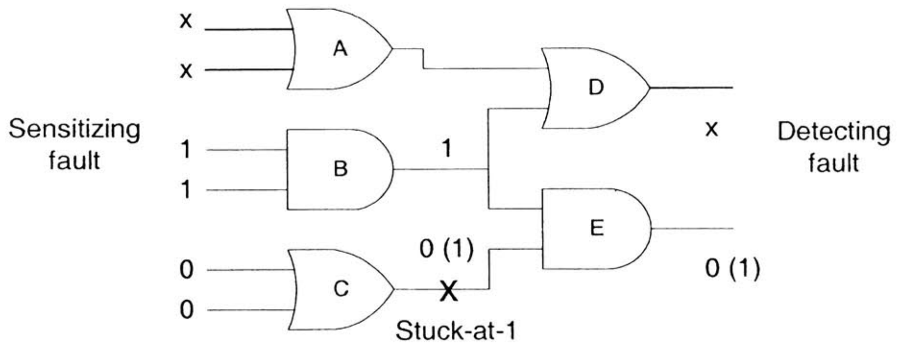


Figure 1-2: *Sensitizing and detecting a stuck-at fault. The number in parentheses is the value generated by the faulty circuit and the other is the value expected from a fault free circuit. An 'x' value indicates that the value can be either '0' or '1' and has no effect on the test.*

The ATPG engine consists of two parts, test generation and fault simulation. The test generator picks a fault from the fault list and creates a vector pair that can sensitize and detect this fault, if actually present. The vector pair consists of an input vector that does the sensitization and an output vector that does the detection. For example, consider the stuck-at fault shown in Figure 1-2. The input vector segment {xx1100} is used to test the fault by doing the following:

1. Propagating a binary '0' to the faulty connection.
2. Propagating a binary '1' to the output of gate B, such that the output of gate C can be observed at the output of gate E. If the output of gate B is a binary "0" the output of AND gate E will be a binary '0', regardless of the output of gate C.

A fault is detected by comparing the actual output of the circuit to the expected output. In our example the expected output vector segment is {x1}. If the stuck-at fault being tested exists in the circuit the actual output segment will be {x0}.

To minimize the test data volume, the ATPG engine tries to test as many faults as possible using a single test vector pair. If many faults are detected in a single vector pair, the number of test patterns needed to test all the faults is reduced and this leads to a reduction in the test data volume. Therefore, the vectors are packed with values that can sensitize and detect as many faults as possible. The number of faults that can be detected in one vector pair is limited by two factors. There is a theoretical maximum for the number of faults that can be detected in a single vector pair because sensitizing and detecting certain faults might require conflicting values in the test vector pair. For example, let us look at the partial vector we generated to detect the stuck-at fault in Figure 1-2. Once the test for that fault is generated, the ATPG engine will pick another fault and try to detect that fault using the same vector pair. Assume that the ATPG engine picked the fault - connection A-D stuck-at-1. To sensitize this fault we need to propagate a binary '0' to both A-D and B-D. However, we cannot propagate a binary '0' to B-D since B-E requires a binary '1' to sensitize the first fault that was picked. Therefore, these two faults cannot be detected simultaneously using the same vector pair. Additionally, since test pattern generation is NP-complete [2], it is impractical to search for all the faults that can be tested using a single vector pair. Therefore, heuristic methods are used to search for a reasonable number of faults that can be detected using the same vector pair. These heuristic methods create a test vector pair as follows:

1. Search the fault list for an untested fault that can be tested using the vector pair.
2. If the search is successful, add the additional test data to the vector pair and mark the fault as tested. If the search is unsuccessful, move on to the next test vector pair.
3. Repeat step 1.

In most cases, the whole input vector is not used to sensitize the faults that are to be detected by the vector [10]. For example, in Figure 1-2 the first two bits were not used to detect the fault. These unspecified bits are randomly filled with zeros or ones and the corresponding values are added to the output vector.

The faults that are detected by a test vector pair are not limited to the faults that were targeted by the vector pair. There might be other faults that are unintentionally sensitized and detected, called random fault detection, by the test values of a given vector pair. Therefore, a simulation engine called a *fault simulator* is used to determine the complete list of faults that are detected by the vector pair. After fault simulation, the faults that can be detected using the vector pair are removed from the fault list and a new vector pair is created using the new fault list. This process continues until all the detectable faults are tested or a sufficient percentage of the faults, as determined by the test engineer, is tested.

The efficiency of a test set is evaluated using the following criteria:

a) Fault coverage

b) Test data volume

c) Test application time

Fault coverage is the percentage of faults that can be detected using the test vector pairs generated by the ATPG engine. Typically test engineers would like to achieve fault coverage of over 99%. Design for Test (DFT) techniques are commonly used to increase the fault coverage of a design. The most commonly used DFT method is called *scan based testing*. The idea of scan based testing is to simplify the test generation process by gaining direct access to the flip-flops of the circuit [2]. This allows the ATPG engine to directly load values into the flip-flops and directly observe the values loaded into the flip-flops. Direct access is achieved by putting a multiplexer in front of the flip-flop, either as a separate element [5] or embedded into the design of the latch [6,7], and introducing extra routing connections to create a chain of flip-flops (shift register). The circuit will then have two modes, a *functional mode* in which the flip-flops behave as they should normally and a *test mode* in which the flip-flops are connected as one or more shift registers. In test mode, the shift registers are called *scan chains* and the flip-flops are called *scan cells*. Figure 1-3 shows an example of a scan design with one scan chain.

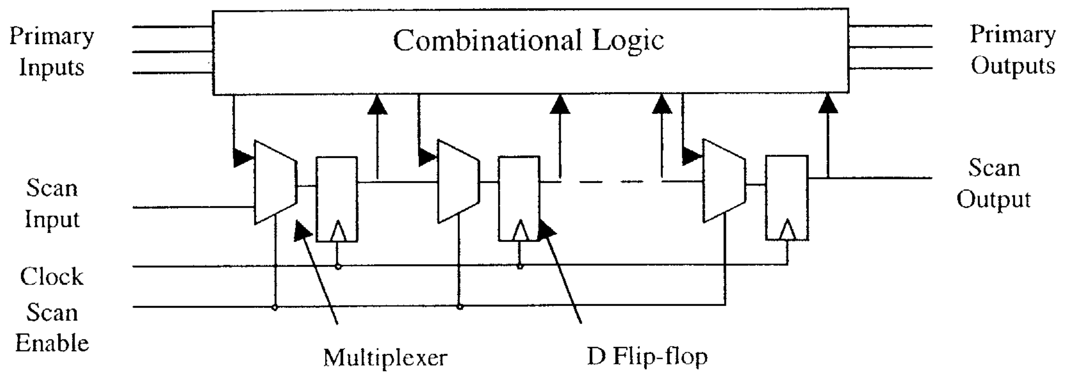


Figure 1-3: *A scan design with one scan chain [2]. The scan input is used to shift in values to the scan chain and the scan output is used to shift them out. The scan enable signal switches the behavior of the flip-flops between scan mode and functional mode.*

The inputs and outputs of the scan chains are considered primary inputs and outputs of the circuit. Therefore, during test mode any value can be loaded into the scan cells by shifting values through the scan chains. Each clock cycle shifts values from one flip-flop to the next flip-flop in the chain. The values in the scan cells can be observed by shifting them out through the scan output. At each clock cycle, the value in the last flip-flop of the chain is shifted out to the scan output. All the scan cells in the circuit can be specified or observed in a number of clock cycles equal to the length of the longest scan chain. A circuit is called *full scan* if all the flip-flops are part of the shift register. If some of the flip-flops are not converted into scan flip-flops, not part of the shift register, then part of the circuit is still sequential and the design is said to be *partial scan*.

To apply a test vector the flip-flops are put in test mode and loaded with test values using the scan input. Once the scan chain is loaded, other test inputs are loaded through the primary inputs. The circuit is clocked in functional mode and the resulting primary output values are observed. Now the flip-flops are once again put in test mode and their values are shifted out through the scan output. Finally, the observed values are compared with the expected values to determine whether the test passes or fails. The ATPG engine creates the testing vectors that are loaded into the scan chains, as described earlier in this discussion.

The overall cost of testing a chip depends on both test application time and test data volume. In scan based testing, the test application time is a function of the number of clock cycles required to test the circuit. Since it takes a number of clock cycles equal to the length of the longest scan chain to load a single scan pattern, the test time depends on the length of the longest scan chain and the number of test patterns needed. The amount of time taken to test a chip determines how many testers are needed to test a batch of chips within the required time. Since testers are very expensive, increased test application times will have a big impact on the test cost.

The test data volume depends on the total number of scan cells and the number of test vectors that are needed to test the circuit. Data volume is important for two reasons. The time required to apply the test data increases with the test data volume. Therefore, the test application cost increases as described above. In addition, ATE's use high performance memory that is very expensive and a large test data volume requires very expensive testers with large memory capacities. Therefore as the memory requirements for the tester increase, the price of the tester increases. Furthermore, the designer might already possess ATE's that were used to test a previous circuit. An increase in the test data volume might make these testers unusable and therefore require new testers to be purchased. All of this leads to an increase in the cost of testing. In the recent past, much work has been done in the area of reducing the cost of testing and many solutions have been put forward. These solutions include simple modifications to the scan based test architecture and more complicated solutions that involve complex DFT. This thesis presents a simple scan based test architecture and corresponding synthesis procedures that can significantly reduce the cost of testing a circuit.

1.3 Reducing the Test Cost

The major factors that determine the cost of testing an integrated circuit are the number of bits loaded into the circuit during testing, *test data volume*, and the time taken to load this data into the circuit, *test application time*. Therefore, recent research [8-15] has focused on lowering test data volume and test application time. Scan chains, the core of the technology that enables scan based ATPG [16], are getting much larger as the designs being tested increase in size and complexity. The data being loaded into test patterns is dominated by the data to be loaded into the scan chains. With relatively few inputs and outputs of the designs being used as terminals for the scan chains, the number of flip-flops per scan chain has also increased dramatically. As a result the time required to operate the scan chains has increased. This leads to an increase in the test application time, because the test application time is dominated by the shift time of the scan chain. To gain an appreciation of the impact of shift time on test application time, consider the typical sequence involved in processing a single scan test pattern.

1. Apply the scan chain configuration.
2. Shift values into the active scan chains.
3. Apply stimulus to the test circuit inputs and observe the outputs.
4. Pulse clock to capture the test circuit response in flip-flops.
5. Shift values out of the active scan chains.

All of these steps— excluding the shift operations in steps 2 and 5 — take one clock period on the tester. The shift operations, however, take as many clock periods as the length of the longest scan chain. For example, a circuit that has 1,000 scan flip-flops in the longest scan chain would spend 2,000 clock cycles shifting values in and out of the scan chains and three clock cycles for the remaining operations. This process can be pipelined by shifting in the next vector while shifting out the previous vector, thereby requiring only half the shift cycles (1,000) per vector. However, the time spent on shifting values to the flip-flops still dominates the test time. Similarly, the test data volume is dominated by the data being loaded into the flip-flops, because steps 1, 3 and 4 require only a few bits of data.

In general, a circuit can consist of one or many scan chains. Consider the traditional scan architecture of Figure 1-4 with *M* scan chains. In this architecture every scan cell can be sensitized and observed through shift operations. To load a value to a given scan cell in a scan chain, the data must be shifted through the preceding scan cells of the chain and shifted out through the subsequent scan cells of the chain. Therefore, even if a test pattern only needs one cell in a scan chain to control or observe values, the whole chain needs to be loaded and unloaded. Hence, a typical test pattern would specify all stimulus and response values for the scan cells.

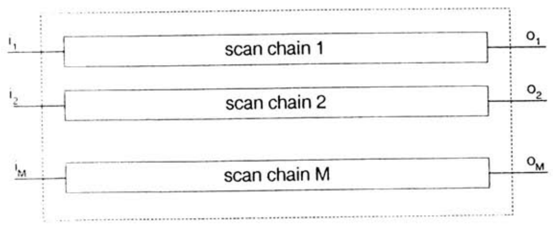


Figure 1-4: *Full scan architecture for traditional scan based ATPG.*

It is commonly known that fault detection requires only a small percent of the stimulus and response points in the design to be accessed for a given test vector [10]. Prior to the focus on test data volume and test application time, the typical practice in the industry was to fill all remaining stimulus points of the pattern (logic X's) with random values to increase random fault detection. However, some new methods being developed have focused on creative ways of treating the randomly filled bits for gains in test data volume and test application time. Although, many test architectures that take advantage of these randomly filled bits have been developed [17-23], none of them have been able to fully utilize the benefits that can be achieved by treating the logic X's differently.

The contribution of this thesis is a new test architecture and associated test generation techniques, which can provide large reductions in both test data volume and test application time. The proposed architecture combines the ideas from two existing techniques that take advantage of the randomly filled bits to create a much more effective test architecture. The two techniques that influence this architecture are bypassing the randomly filled bits [10,23] and parallel loading of scan chains [8,9,17]. A detailed description of these two techniques is given in the following sections. In [10], bypassing the logic-X's is achieved using a scan chain reconfiguration technology that can move scan segments into and out of the active scan chains. Parallel loading of scan chains with the same values, uses common scan inputs to allow for the logic-X's of the scan cells to be filled as those in other scan cells and thereby reduce the data volume [8]. This thesis focuses on combining the ideas of these two architectures to create a reconfigurable parallel scan architecture that can achieve much larger reductions than either of the above mentioned methods on its own.

1.4 Related Work

Much research has been done in the area of reducing the cost of testing integrated circuits. The cost of testing a circuit is increasing relative to the total cost of design and manufacturing [1]. Therefore, creating low cost test strategies has become an important research area. Scan based testing is the most popular test strategy because it provides high fault coverage and simplifies test pattern generation [2]. However, the scan chains introduced in scan based testing increase the test data volume and test application time. Research on low cost test solutions comprises of architectures that reduce the overhead of scan based testing, architectures that mix scan based testing with other methods and non-scan architectures. This section describes some of these low cost test solutions.

In *sequential testing*, only some of the scan cells are connected to the scan chain and the design is partial scan (not full scan and not combinational in test mode). Many hybrid test generation schemes [18-20], which use both scan based testing and sequential testing, have been proposed to reduce test application time. Since these schemes are not full scan, multiple clock cycles are required to propagate a value from the input to the output. The number of clock cycles required depends on the longest sequential path in the test. The simulation required to generate test patterns for large sequential circuits is very complicated and time consuming. Therefore, these hybrid test schemes do not scale well and cannot be used with large sequential circuits. There is no evidence of the hybrid test methods being tested on any circuits with more than three thousand gates [18-20].

Another proposed solution is to create *multiple scan chains* [19,22] and load them in parallel using one input per scan chain. This reduces the length of the longest scan chain and thereby decreases the test application time. However, loading a large number of scan chains in parallel requires a large number of input and output pins on the chip, which can be impractical because the number of I/O pins is physically limited by the size of the chip. Therefore, the number of pins that are available limits the parallelism that can be achieved. Furthermore, in both of the above-mentioned schemes, gains are limited to test application time while test data volume is not addressed.

In *Partial Parallel Scan* [21], the test architecture allows for groups of flip-flops to be changed from scan flip-flops to non-scan flip-flops during the testing process. This allows the test engineer to switch between different levels of partial scan and potentially save the time and data spent on loading unnecessary scan cells. However, complex control logic that requires a high hardware overhead is needed to facilitate this switching architecture. Partial Parallel Scan is able to reduce test application time by one to two orders of magnitude [21]. Although the results are good, this architecture has a few drawbacks. Since this is not a full scan technique, the test generation process becomes much harder for the ATPG engine and results in lower test coverage. In addition, even though partial scan is used to minimize the hardware overhead, the extra 6%-19% area overhead of this DFT architecture [21] is large, and therefore impractical to use in many designs. Another drawback of this method is that it only focuses on reducing the test application time and does not make an effort to reduce the test data volume.

Another method that can avoid the loading of unnecessary scan cells is to allow the ATPG engine to bypass flip-flops that are not used in a given vector [10,23]. *Dynamic Scan* [10] does this by breaking each scan chain into scan groups and bypassing unused groups using a simple configuration of multiplexers. This method of bypassing unused cells comes with a low hardware overhead since only a few multiplexers are used. Scan cells are grouped together based on the frequency of usage as determined by the ATPG engine. The ATPG engine then specifies which scan groups are needed for each test vector. Bypassing the scan cells that are not needed saves both test data volume and test application time. The reductions that can be achieved are approximately 70% of the test data volume and test application time, of regular scan based testing [10]. This architecture provides a very elegant and simple method of reducing the cost of test, but the gains achieved are much less than what can be achieved by the architecture proposed in this thesis.

Built in Self-Test (BIST) techniques use Linear Feedback Shift Registers (LFSR) to generate the test patterns [2]. These LFSR's are built around the circuit so that an ATE is not needed to input these test vectors. The test data volume is significantly reduced since most of the data no longer needs to be fed into the chip. The test vectors created by a LFSR are pseudo-random sequences of binary values based on an input seed given to the LFSR. These vectors are not created by targeting faults in the circuit like an ATPG engine does. Therefore, the on chip test generation is dependent on random detection of faults and is much less efficient than test vectors created by an ATPG engine. Due to this inefficiency, the number of test vectors (LFSR sensitizations in BIST) increases by as much as ten times and increases the test application time [4]. The most significant gains in test application time have been shown using logic BIST [25,26,29,30] and deterministic BIST [12,24,28], both of which are hybrid schemes between ATPG and BIST. However, these schemes come at a hardware overhead of 13% to 20% [1] and require certain modifications to the non-DFT elements of the circuit. These modifications can be intrusive to the functionality of the circuit and might not even be possible in certain designs. Even though such drawbacks exist, BIST based test methods are still very popular, since the use of expensive ATE time is avoided in these methods.

A recently proposed scheme, *Illinois Scan* [8], provides an elegant solution to the low cost test problem. In Illinois Scan, a large number of scan chains are grouped into a few scan groups and loaded in parallel using one input pin per scan group. Illinois Scan consists of two modes in which the architecture can operate. The first mode, called the *broadcast mode*, connects each group of scan chains to one input pin. Therefore, a single test vector can be broadcast to all the scan chains that are connected in parallel. However, by connecting many chains to one input, new dependencies are added to the system. Any two scan cells in the same row of different scan chains in the same group will always have the same value. Therefore, certain tests that require different values in the same row of the scan chains cannot be applied to the circuit. To solve this problem a second mode called the *serial mode* is maintained. In this mode, all the scan cells are connected together as one long scan chain. This architecture performs well, as long as a large percentage of the vectors can be run in broadcast mode, since serial mode patterns are equivalent to regular scan testing. However, as the number of scan chains loaded in parallel, known as the *parallelism*, is increased, the number of dependencies in broadcast mode increases. This causes the number of faults being detected in that mode to decrease, which in turn increases the number of serial mode vectors. Therefore, this architecture is limited by the inability to detect most faults in broadcast mode when a large number of scan chains are loaded in parallel. A variation of the scheme with two broadcast modes [9] was proposed to reduce the number of serial patterns needed. However, expensive serial mode patterns are still needed when the parallelism is high and these patterns overshadow the gains achieved during broadcast mode. This scheme is very appealing due to its simplicity and low hardware overhead, but cannot compete with the BIST based solutions unless serial mode patterns can be eliminated. The solution presented in this thesis is based on the idea of eliminating the need for a serial mode, while allowing for a high degree of parallelism.

Chapter 2

A Reconfigurable Shared Scan-In Architecture

A new architecture for testing integrated circuits is proposed in this chapter. We call this architecture a Reconfigurable Shared Scan-In Architecture (RSSA). The architecture is introduced by giving a detailed description of the existing technologies it is based on, and by showing how greater benefits can be achieved using the new test architecture. The concept for RSSA builds on two existing methods, sharing scan inputs [8,9,17] and Dynamic Scan [10].

Figure 2-1 shows the shared scan-in architecture commonly known as Illinois Scan. This architecture operates in two modes. Assume that the circuit has m scan inputs and N scan chains. In the first mode, M groups of N/M scan chains are loaded in parallel using the M scan inputs. This mode of operation is called the broadcast mode because the same input vector is broadcast to many chains. In the second mode, called the serial mode, the scan chains behave as in regular scan, with each group of N/M scan chains being connected together as a single scan chain. We will focus on the broadcast mode of Illinois Scan for now. Since many scan chains share the same input in broadcast mode, it is possible to have more scan chains than the number of scan inputs available. Being able to increase the number of scan chains reduces the length of each scan chain. In Figure 2-1 the parallel loading reduces the length of each scan chain by a factor of N/M. Since the time taken to apply a test vector is proportional to the length of the longest scan chain, reducing the length of the scan chains by a factor of N/M reduces the time required to apply one test vector by a factor of N/M. If the number of test vectors applied does not change[4], Illinois Scan should reduce test application time by a factor of N/M compared to regular scan testing.

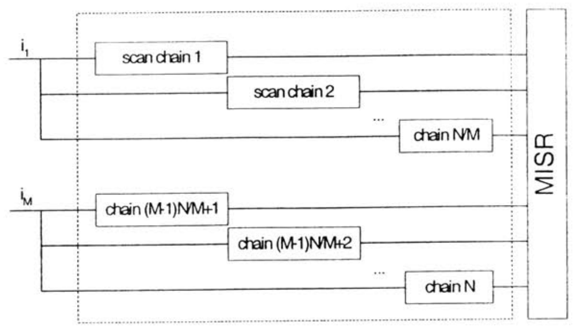


***Figure 2-1**: The Illinois Scan architecture where M input pins feed N scan chains. Each of the M scan chains from Figure 1-4 are broken into N/M smaller scan chains and are loaded in parallel.*

On the output side, a Multiple Input Signature Register (MISR) [8] is used to test whether the resulting output is the expected one. The MISR works by taking all the output values from the N chains and creating a signature for that output. The length of the signature is a function of the total number of bits being compacted. A circuit with more flip-flops will require a larger signature to minimize the probability of two outputs getting the same signature and a faulty test being passed. Since this architecture uses a MISR signature to check whether a test is successful, we do not use output vectors. Therefore, in the rest of this thesis we will refer to an input vector simply as a vector.

[4] We will show later that the number of test vectors needed to test the circuit actually increases because the constraints of parallel loading reduce the number of faults that can be targeted in each test vector.

Since a group of *N/M* scan chains are loaded in parallel using one scan input pin, each of these scan chains will have identical values loaded in them for each test vector. For example, if a test vector {0,0,0,1,1,1,0,1,0} is loaded using a scan input *i*, then all *N/M* scan chains that share input *i* will load the same test vector {0,0,0,1,1,1,0,1,0}. In general, fewer faults can be targeted with one test vector, resulting in an increase of the number of vectors needed. Furthermore, let us assume that testing a certain fault requires the first value in chain 1 to be a 0 and the first value of chain 2 to be a 1. It would not be possible to test this fault using the setup that has been described so far. In the method described the first value of all *N* chains has to be the same. To detect such faults that cannot be detected in this parallel configuration, the serial mode is used. In serial mode, each group of *N/M* parallel scan chains is connected together and can be treated as one regular scan chain.

Illinois Scan works by first testing as many faults as possible using the broadcast mode and then testing the remaining faults using the serial mode. The test data volume and test application time can only be reduced when operating in broadcast mode since serial mode is equivalent to regular scan testing. Therefore, the effectiveness of Illinois Scan depends on the number of faults that can be detected in broadcast mode.

The gains of Illinois Scan depend on the percentage of vectors applied using broadcast mode and the number of scan chains that are connected in parallel. Lets assume that *M* groups of *N/M* scan chains are loaded in parallel during broadcast mode. The test data volume and test application time reductions are governed by Amdahl's Law, which calculates the speedup achieved by any parallel process. Let us consider the test application time reduction of a circuit being tested using Illinois Scan. The faults that cannot be detected in broadcast mode must be detected using the serial mode. The test vectors that are applied in serial mode are called the *strictly serial* part of the process because this work cannot be done in parallel. Amdahl's Law, when applied to parallel loading of scan chains, states the following. If K chains are loaded in parallel and B percent of the process is strictly serial, then the test application time reduction, the *speedup*, is:

$$\text{Speedup} = \frac{K}{(B*K) + (1-B)}$$

If V_b is the number of test vectors applied in broadcast mode and V_s is the number of test vectors applied in serial mode, the percentage of the work that is strictly serial (B) is:

$$B = V_s / (V_b + V_s)$$

The value of B for data volume reduction is also the same as above, since the strictly serial part is once again the data that needs to be loaded in serial mode. Therefore, the following analysis that focuses on test application time is applicable to the test data volume as well.

In the relationship between B, V_s and V_b, an increase in V_s will increase the value of B. In the equation for speedup, an increase in B will always increase the denominator of the equation, since $K > 1$ and $0 < B < 1$. An increase in B will decrease the speedup achieved. Therefore, an increase in V_s will reduce the speedup. Amdahl's law reinforces our claim that the benefits of Illinois Scan are reduced as the number of vectors loaded using serial mode increases.

The equation for speedup also shows that the speedup increases when K increases, because $0 < B < 1$ and the numerator grows faster than the denominator. Since K is the number of scan chains that are loaded in parallel, increasing the parallelism will give greater reductions in the test application time.

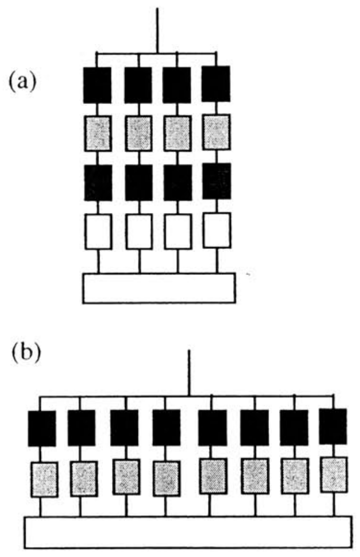


Figure 2-2*: Dependencies between scan cells increase as the sharing increases. In Figure 3(a) four chains share the same input and four scan cells in each row must have the same values. When the sharing increases to eight scan chains, eight scan cells are in the same row and have the same values.*

Figure 2-2 shows two examples of a circuit with different values of K. In Figure 2-2(a), K equals 4 and four groups of four scan cells each must have the same value loaded. In

Figure 2-2(b), *K* equals 8 and two groups of eight scan cells each must have the same value loaded. Cells shaded in the same color take on the same value. In general, however, as more scan chains are loaded in parallel, more dependencies are created in the scan chains. As the number of dependencies in the scan chains increase, the probability of some fault being undetectable increases, because fewer combinations of values can be loaded into the chains. When increasing the parallelism, an increasing number of faults that cannot be tested using the broadcast mode will have to be tested using the serial mode. However, testing more faults in serial mode reduces the gains of this architecture. On one hand, increasing the parallelism increases gains. However, on the other hand, increasing the parallelism beyond a certain threshold increases the number of faults that need to be detected in serial mode and thereby reduces the gains.

When the parallelism is low, increasing *K* does not create many untestable faults and the gains achieved by parallel loading outweigh the overhead of the extra serial mode vectors. However, as the parallelism becomes large, the number of faults that need to be detected in serial mode increases dramatically and at some point the gains achieved by loading the data in parallel are overshadowed by the increase in serial mode vectors. The effectiveness of Illinois Scan has been somewhat improved by creating two broadcast modes that use different scan configurations [9]. This allows some faults that were not detected in the first broadcast configuration to be detected in the second broadcast configuration, since the two configurations have different dependencies. However, this improvement fails to increase the maximum reductions of Illinois Scan by more than a factor of two and in most cases is much less [9].

The goal of this thesis is to create an architecture that can eliminate the need for serial mode testing by making it possible to load the scan chains in all possible broadcast configurations. This allows us to increase the parallelism unless a given fault is not testable in any of all possible broadcast modes. To achieve the goal of detecting all the faults in broadcast mode, a simple yet flexible reconfiguration architecture is needed.

The motivation for our reconfiguration architecture comes from Dynamic Scan [10], where each scan chain is broken into groups that can be either loaded or skipped during the application of a given test vector. In Dynamic Scan, a simple configuration of multiplexers is used to either load or bypass each scan group. Similarly, the proposed reconfiguration architecture uses a simple configuration of multiplexers to select the scan input that drives each scan chain. The scan input that drives each scan chain is determined during the ATPG process. Therefore, the configurations are created during the ATPG process as needed for detecting all faults. This is much more efficient than determining the configurations prior to ATPG, since the ATPG engine has the best knowledge of what values need to be set to detect a fault.

We now present the Reconfigurable Shared Scan-In Architecture (RSSA) proposed in this thesis. The architecture, which is shown in Figure 2-3, allows the scan chains to be connected to the scan inputs in multiple configurations. A given scan chain can be connected to any of the scan inputs by applying the corresponding control signal to the multiplexer that is connected to the scan chain.

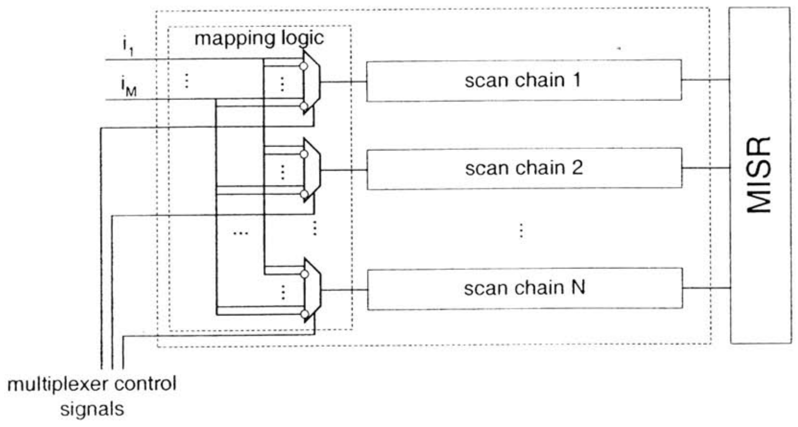


***Figure 2-3**: The new architecture allows multiple scan configurations that can be used during testing. In each configuration a different set of scan chains is driven by a given input. The multiplexer in front of each scan chain is used to select the inputs that drive the scan chain.*

The mapping logic consists of $2M$-way multiplexers at the beginning of each scan chain and allows the scan chain to be driven by any of the M inputs depending on the control signal to the multiplexer. The control signal to each $2M$-way multiplexer is $\log(2M)$ bits wide. Figure 2-4 shows how each scan chain is connected to all inputs. In addition to connecting a scan chain to all the inputs, we also facilitate a connection to the inverted values of each input. This is important because it gives the architecture a larger set of possible scan chain configurations. The multiplexer control signals are supplied through the primary inputs of the circuit. Since there are N multiplexers with $2M$ inputs each, $N\log(2M)$ primary inputs are required to control them. However, primary inputs of the circuit are limited and this would be an impractical solution. Therefore, in Section 3.4 we present an optimization that reduces the number of primary inputs needed to just one.

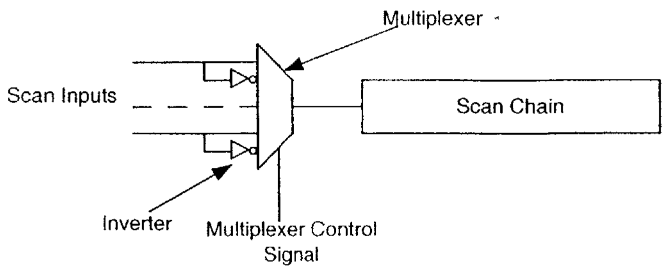
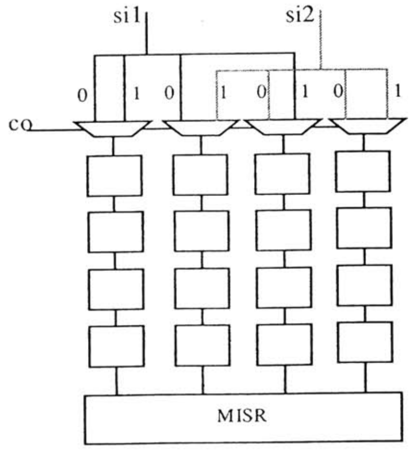


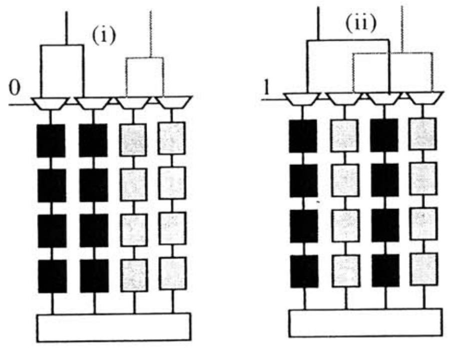
***Figure 2-4**: The scan chain can be driven by one of many scan inputs. The input corresponding to a given test vector is connected to the scan chain using a multiplexer. The scan chain can be connected to any of the connected inputs or its inverted value.*

The flexibility of this architecture is further increased by the ability to change configurations while shifting in a test vector[5]. This can be achieved by simply changing the multiplexer control signal before the shift operation that requires the new configuration. The following example shows the added flexibility achieved by changing configurations while shifting in a test vector. The design in Figure 2-5 has sixteen scan cells, four scan chains and two scan inputs. The scan cells in Figures 2-5(b) and 2-5(c) are color coded to show their relationship to the scan inputs. The values shifted through the first scan input are shown in black and the values shifted through the second scan input are shown in gray. For simplicity, let us assume that the multiplexer control signal can only select between two configurations. A control signal of zero selects the configuration in Figure 2-5(b)-(i) and a control signal of one selects the configuration in Figure 2-5(b)-(ii). If the multiplexer signal is constant during the shifting process, only the two configurations in Figure 2-5(b) can be used to shift values to the scan cells.

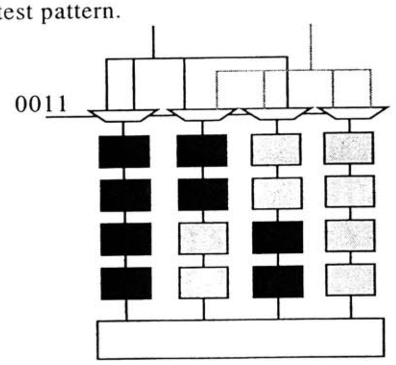
[5] Current ATPG engines are not able to create test vectors with different configurations for each shift cycle. Therefore, ATPG is run with no constraints and post ATPG processing is used to determine the configuration for each shift cycle. This is explained in Section 3.3.



(a) 4 scan chains connected with two configurations.



(b) The two configurations available to scan data with static control for the test pattern.



(c) Dynamic reconfiguration of chains during a single test pattern.

Figure 2-5*: Example reconfiguration structure and configurations. Shading is used to depict the connection between scan inputs and the values in the scan flip-flops.*

However, if the multiplexer signal can be changed during the shifting process, we can pick either configuration for each row of scan cells. This increases the number of possible configurations from two to sixteen, since each of the four rows can have two different configurations. An example of such a configuration is shown in Figure 2-5(c). In this configuration the multiplexer signal is set at zero for the first two shift cycles and set at one for the last two shift cycles.

RSSA allows for the multiplexer control signal to be kept constant or to be changed while shifting in a test vector. If the control signal is kept constant during the shifting of test vector, the configuration is called a *static configuration* and if the control signal is changed during the shifting of a test vector, the configuration is called a *dynamic configuration*.

In this thesis, we present two synthesis procedures that take advantage of the proposed architecture, RSSA. In one solution, the control signal is not changed during the shifting of test vectors and the configurations are always static. In the other solution, the control signal is changed during the shifting of some vectors and dynamic configurations are generated during the shifting of these vectors. It should be noted that the implementation of dynamic configurations requires that the timing of the control signal be carefully adjusted to match the shift operation.

In this chapter we presented the reconfigurable shared scan-in architecture, RSSA. The motivation for RSSA was described by identifying the deficiencies of Illinois Scan. We proposed to eliminate these deficiencies using RSSA, which has a high degree of flexibility, since it can create a large number of parallel configurations. In comparison, Illinois Scan has only one parallel configuration. The ability to create a large number of configurations increases the possibility of detecting all the faults in broadcast mode and makes it possible to achieve a parallelism much higher than that of Illinois Scan. Avoiding the use of serial mode and the high level of parallelism provides the potential for a large reduction in the test cost. The next section presents the two synthesis procedures that facilitate the use of the proposed architecture.

Chapter 3

Synthesis Procedure

The synthesis procedure for RSSA is a multi-part process. First, the scan chains, which are chains of flip-flops that are used to load test data, are created, then a series of test configurations that are used to apply the test data is selected, and finally the logic that connects the scan chains to the scan inputs is optimized. The scan cells of the circuit are grouped into scan chains based on the topology of the circuit. This process is described in Section 3.1. Once the scan chains have been created, the test engineer has the option of two different synthesis solutions that connect the scan chains to the scan inputs. In one solution, given in Section 3.2, an analysis engine is used to determine the input that drives each scan chain. In the other solution, given in Section 3.3, each configuration corresponds to a different prime number M and every M^{th} scan chain is connected to the same scan input. Finally, in Section 3.4, we present an optimization, based on the fact that only a small number of all the possible configurations are actually used, to reduce the hardware overhead of the logic that connects the scan chains to the scan inputs.

3.1. Creating the Scan Chains

This section describes the algorithm used to create the scan chains of the circuit under test. The algorithm uses topological information from the circuit when creating the scan chains. The *cone of logic* of a given output is defined to be all the inputs that drive the output and all the logic in-between the inputs and the output. Figure 3-1 shows the cones of logic for four outputs in a circuit. These outputs can be either scan cells or primary outputs. The cone of logic of each output of the circuit is used to determine which scan cells are grouped together.

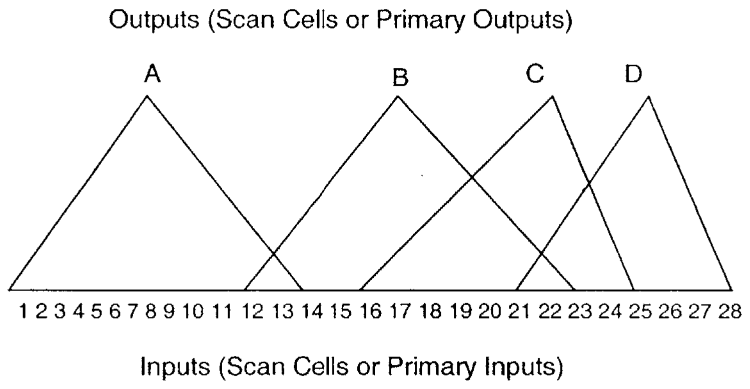


Figure 3-1: *The cones of logic of a circuit segment. The peak of each cone is a scan cell that is influenced by all the logic within the cone. The base of each cone is the entire set of scan cells that influence the cone.*

A *cone conflict* occurs when a pair of scan cells belonging to the same cone of logic are in two different scan chains. For example, a cone conflict will occur if any of the scan cells 1-13 from Figure 3-1 are not in the same scan chain. Every stuck-at fault is part of one or more cones of logic and the only scan cells that are needed to sensitize a fault are the scan cells at the base of those cones of logic. For example, any fault that is in the logic cone ending at scan cell A will only require the scan cells 1-13 to sensitize the fault. It should also be noted that not all of these scan cells are always needed to sensitize the fault. If all the scan cells needed to test a fault are in the same scan chain, a conflict cannot occur. A cone conflict between two scan cells is only a potential conflict unless the following additional criteria are satisfied.

1. Values needed in the two scan cells are not compatible (logic-0 and logic-1). If the values needed are the same, sharing the same input is not a problem since the same data needs to be loaded in both scan cells
2. The two scan cells are in the same shift position relative to the scan input. If the scan cells are not in the same row, the data that is needed for the two scan cells are loaded during different shift operations. For example, a value needed in the third row is loaded one clock cycle before a value needed in the second row.
3. The two scan cells are in scan chains that are sharing the same scan input. If the scan cells do not share the same input, they do not have to share the same data.

Thus the cone of logic is used as a simple mechanism for constructing the scan chains.

The scan chain creation algorithm works by trying to minimize the number of cone conflicts between the scan chains that are created. Assume that we are constructing the architecture for a design with F scan cells and we chose to have N scan chains. Then the length of all scan chains is no more than L[6], where $L = \lceil F / N \rceil$. Cones of logic are

[6] We are assuming that all the scan chains will use a single scan clock. If multiple scan clocks are used, the scan chains driven by each clock might have a different value of L.

extracted from the circuit description for every observable point (scan cell or primary output) of the design and sorted by size. Starting with the inputs of the largest cone in the list created, the first L unassigned scan cells are assigned to create the first scan chain. The following L cells are assigned to the next scan chain and the process continues until all scan cells are assigned to some scan chain. Since the scan cells in the same cone of logic are grouped together, it is likely that most scan cells in a given cone are either in the same scan chain or adjacent scan chains. Therefore, a majority of the scan cells that need to be specified to detect a given fault are either in the same scan chain or adjacent scan chains. For example, in Figure 3-2 the scan cells from Figure 3-1 have been grouped into four scan chains. Each scan chain has seven scan cells. All the cells belonging to cone A are in chains 1 and 2, those of cone B are in chains 2,3 and 4, of cone C are in chains 3 and 4 and the cells of cone D are in chain 4. Since this is only part of the circuit and the complete circuit will have many more scan chains, the cone conflicts between scan chains occur only in a very small number of adjoining scan chains.

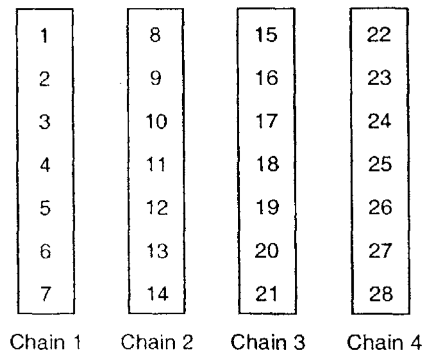


Figure 3-2: *The scan cells from Figure 3-1 being grouped into four scan chains using the algorithm described above.*

The scan chain creation algorithm takes the circuit prior to test logic insertion and creates *N* groups of scan cells. To keep the degree of freedom for DFT synthesis as high as possible, the order of scan cells in each scan chain is not specified. In this way, the DFT synthesis engine has a higher flexibility to determine the order of each scan chain such that physical and routing constraints of the circuit can be satisfied. It should be noted that an alternative solution that specifies the exact scan chain ordering would increase the number of detected faults. However, fixing the scan chains without considering physical and routing constraints may not yield routable designs and is unacceptable from a synthesis standpoint [32]. Therefore, the groups of scan cells created by the selection algorithm are sent to the synthesis engine that creates the actual scan chains.

3.2 Compatibility Based Reconfiguration

Compatibility based reconfiguration is one of the two solutions that we propose for selecting the scan inputs that drive each scan chain. In compatibility based reconfiguration, scan configurations are created using an iterative process until the percentage of detected stuck-at faults, known as fault coverage, reaches a minimum value set by the test engineer. The iterative process works by creating a configuration, determining the testable faults in that configuration and then creating another configuration to test the faults that have not been tested yet. This process is repeated until the required fault coverage is reached.

3.2.1 Creating the First Configuration

The *first configuration* is the configuration that the ATPG engine uses first to load test patterns into the circuit. In compatibility based reconfiguration, the process used to create the first configuration, differs from the process used to create the rest of the configurations. This section describes the method used to create the first configuration. Let M denote the number of primary scan inputs available. As described in the previous section, a majority of the scan cells that are dependent on each other are, either in the same scan chain or in adjacent scan chains. Therefore, the first configuration is obtained by assigning M groups of every M^{th} scan chain to the same input. The scan chains [1, M+1, 2M+1, ...] are connected to the first scan input, the scan chains [2, M+2, 2M+2, ...] are connected to the second scan input and the scan chains [M, 2M, 3M, ...] are connected to the M^{th} scan input. More formally, let N be the number of scan chains, i be a scan input, where $1 < i < M$ and j be a number such that $0 < j < N/M$. In the first configuration, the scan chains $jM + i$ for all j are connected to the i^{th} scan input. To understand the effectiveness of this scheme, consider the following. The scan chains created in Figure 3-2 are connected to two scan inputs as shown in Figure 3-3. The cone conflicts of these scan cells, which are given in Figure 3-1, show that a potential conflict can only occur between chains 2 and 4 and only when testing a fault in cone B, because chains 2 and 4 are the only chains that have scan cells from the same cone and share the same scan input.

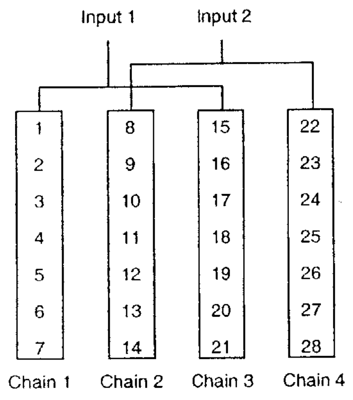


Figure 3-3: *The scan chains from figure 3-2 being connected to two scan inputs in the first configuration.*

The output of each scan chain is sent to a MISR as in Illinois Scan and the signature created by the MISR[7] is used to test whether a test vector passes or fails. Once the first configuration is selected, ATPG is run to detect as many faults as possible in this configuration and the remaining undetected faults are determined. The next step is to

[7] See Section 3.1

create the additional configurations that are required to detect the remaining undetected faults. These additional configurations are created using the theory of compatibility analysis presented in [33] and described below.

3.2.2 The Theory and Application of Compatibility Analysis

In this section, we describe the theory of compatibility analysis and how it can be applied to determine the scan chains that should be driven by each scan input. Compatibility analysis is used when creating configurations after the first configuration. In compatibility analysis, two scan chains are said to be *directly compatible* if they can be connected to the same scan input without creating a constraint that prevents an otherwise detectable fault from being detected. Similarly, two inputs are said to be *inversely compatible* if they can be connected to the same scan input with one chain getting the inverted signal of the input without creating a constraint that prevents a fault from being detected. A set of scan chains forms a *compatibility class*, if all the scan chains in the set are directly or inversely compatible to each other. The same scan input could be used to drive all the scan chains that are part of a compatibility class without introducing any undetectable faults. To detect all the faults, we need a scan input per compatibility class. However, there might be more compatibility classes than the number of scan inputs that are available. Therefore, more than one compatibility class might have to be connected to the same scan input creating some undetectable faults. The idea of a *composite compatibility relation* [36] is used to solve this problem.

The composite compatibility relation is defined as follows. Assume that K different configurations are used and that λ_i is the set of undetectable faults introduced by configuration *i*. Then the K configurations define a composite compatibility relation if and only if $\lambda_1 \cap \lambda_2 \cap \ldots \cap \lambda_K = \varnothing$. In simpler terms, the faults that are undetectable in one configuration are detected in some other configuration and the K configurations together detect all the faults.

To create the additional configurations using the above mentioned compatibility relations, we need a method of determining whether two scan chains are directly or inversely compatible. In general, there are two strategies of doing this: passive and active. The *passive strategy* [33] takes a set of pre-computed test patterns and two scan chains, and determines whether all the test patterns of the pre-computed test set can be applied if the two chains are connected to the same input. In the passive strategy it is possible that two scan chains that are indeed compatible be diagnosed as being incompatible, because a pre-computed test set contains only one of many different tests that can be applied to detect a given fault of the circuit. For some test patterns that cannot be applied when the two scan chains are connected to the same input, there might be other patterns that detect the same faults. For example, Figure 3-4 shows two input sets that can be used to detect the same fault. In a pre-computed test set the ATPG engine will pick one of these two options to detect the fault. Lets assume that the ATPG engine picked the input set given in Figure 3-4(a) and that the second input to gate B cannot be set to logic 1 when the two scan chains picked above are connected to the same input. In this case, the fault can still be detected by setting the first input of gate B to be logic 1.

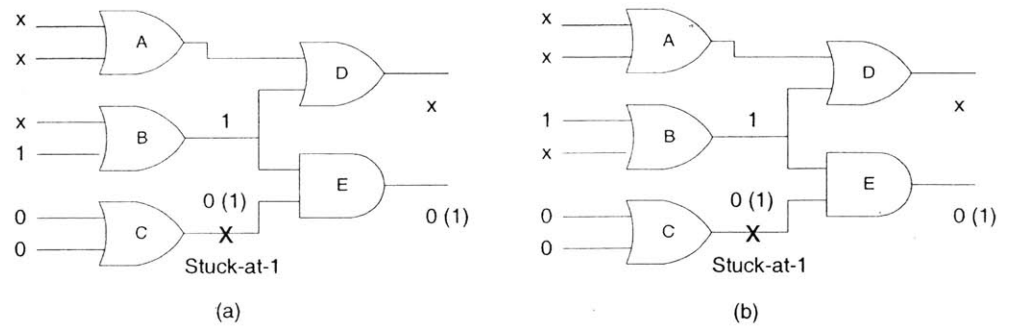


Figure 3-4: *Two methods of propagating the same stuck-at fault to an output. The output of gate B has to be logic 1 for the stuck-at-1 to be propagated to the output. This can be achieved by setting the second input of gate B to logic 1 (Figure 3-4(a)) or setting the first input of gate B to be logic 1 (Figure 3-4(b)).*

The *active strategy* [34-36] for determining compatibility between two scan chains works as follows. Two scan chains are constrained to have the same values and then the ATPG engine is run to check whether all the faults can be detected. This strategy gives an accurate diagnosis of whether two scan chains are compatible, because the ATPG engine will examine all the possible inputs for detecting a fault. However, the processing time of the active strategy is much higher because the analysis may involve hundreds of ATPG runs. Therefore, the processing time of compatibility analysis becomes an important issue that determines the applicability of the proposed synthesis procedure in situations where test generation time is critical. This process relies on detecting a large number of faults in the first configuration so that the many ATPG runs needed during compatibility analysis are performed on a much smaller fault list.

3.2.3 Using Compatibility Analysis to Create Configurations

If the required fault coverage is not achieved after the first configuration, compatibility analysis is used to create more configurations. A scan chain is said to be *independent* if it does not belong to any compatibility class. The *active fault list*, the fault list used by the ATPG engine, is set to contain all faults that are not detected in the first configuration, and the next configuration is created using the following three-phase process:

1) Deriving passive compatibility classes: Initially, all the scan chains of the circuit are independent. The scan chains are grouped into a maximum of M compatibility classes, where M is the number of scan inputs, using the passive strategy with a pre-computed set of test patterns. The scan chains that do not fit in these M compatibility classes are still independent.
2) Reducing independent chains using the active strategy: Active compatibility analysis is run on all the remaining independent scan chains to determine whether they fit in any of the M compatibility classes. If an independent scan chain is either directly or inversely compatible to all scan chains in a compatibility class, then the scan chain is included in this compatibility class.
3) Minimizing number of untested faults: Adding the remaining independent scan chains to the M compatibility classes will make some faults undetectable. The goal of this step is to minimize the number of faults that are made undetectable. Each independent scan chain is compared against all the compatibility classes and added to the group that creates the least number of undetectable faults. This phase continues until all independent scan chains are included into a compatibility class.

This process might be halted after the first or second phase if all the scan chains are already part of the M compatibility classes. If the process is halted early, this will be the last configuration, since all the scan chains are part of a compatibility class and all the faults can be detected. If the process had to go through the third phase, ATPG is run again and the untested faults are determined. The fault list is now set to be these untested faults. To detect these faults, another configuration is created by repeating the three phases described above. The process is repeated until one of the following conditions is met:

1. All detectable faults have been tested.
2. The required fault coverage has been reached.
3. A maximum number of configurations, as determined by the test engineer, have been reached.

The worst-case complexity of this procedure is $O(MN')$ ATPG runs, where M is the number of scan inputs and N' is the number of independent scan chains after the passive phases. Using the passive strategy in phase one to reduce the number of independent scan chains reduces the work that needs to be done by the computationally expensive active strategy. However, the compatibility analysis could still be time consuming if the number of faults in the fault list is large since many scan chains may remain independent after the first phase. Therefore, in the next section we describe an alternative reconfiguration process that has no extra ATPG runtime cost.

3.3 Prime Based Reconfiguration

Prime based reconfiguration is an alternative synthesis process to compatibility based reconfiguration, for assigning the scan chains to scan inputs. Prime based reconfiguration does not use an analysis engine to create configurations and therefore is not as computationally intensive as the compatibility based scheme described in the previous section. In this method, we create a new configuration by increasing the number of scan input pins (*M*) that are used to load the test data. As shown in Figure 3-5, fewer scan chains share the same input pin and therefore, the same data, when *M* is increased. Therefore, increasing the number of input pins reduces the dependencies of parallel loading. New configurations are created until all faults can be detected in the available configurations. Dynamic configurations[8] are used to minimize the number of input pins that are needed.

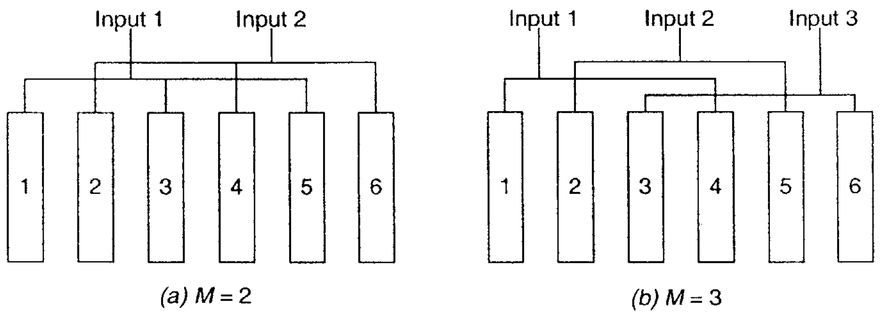


***Figure** 3-5: The shared scan-in configurations for M=2 and M=3. When M=2 two sets of three scan chains each share the same data. However, when M is increased to 3, three sets of only two scan chains each share the same data.*

[8] See Section 3.2

3.3.1 ATPG for Dynamic Configurations

In prime based reconfiguration, the scan inputs used to load each scan chain might change during the process of loading a test vector. We call such configurations dynamic configurations. In general, static configurations are used and the scan inputs used to load each scan chain are only changed in between the loading of test vectors. The ATPG process for a dynamic configuration is different from that for a static configuration. When using a static configuration, the ATPG engine is given the constraints introduces by the selected configuration, and therefore, creates test vectors within these constraints. However, when using dynamic configurations there are multiple constraints, because the configuration can change after each shift cycle and each configuration represents a different set of constraints. Since we can use any of the multiple configurations at each shift cycle, we only need to satisfy one set of constraints for the values loaded in a given shift cycle. Therefore, ideally we would like the ATPG engine to take these multiple constraints and create test vectors that satisfy any one of these constraints during each shift cycle. Unfortunately, current ATPG technology cannot handle ATPG with multiple constraints.

Since this ideal solution is not possible given current ATPG technology, we use a less effective alternative. In the alternative method, ATPG is first run assuming that all the scan chains can be loaded independently. After ATPG is run, we use a post-processing method to look at the values loaded during each shift cycle and pick a configuration that can load the values required during this shift cycle. For each shift cycle, this process iterates through all possible configurations until a configuration that can load the test data for the current shift cycle is found. For example, consider the circuit in Figure 3-6(a). The circuit has two configurations that can be selected using a control signal of zero or one. Assume that the following ATPG test vector needs to be loaded into the scan chains.

Scan chain 1: {1,1,1,1}
Scan chain 2: {1,1,0,0}
Scan chain 3: {0,0,1,1}
Scan chain 4: {0,0,0,0}

This test vector can be loaded into the circuit as shown in Figure 3-6(b). During the first and second shift cycles, a control signal of zero connects scan chains 1, 2 to scan input one and scan chains 3, 4 to scan input two. Therefore, the scan chains 1, 2 and 3, 4 contain the same data. During the third and fourth shift cycles, a control signal of one connects scan chains 1,3 to scan input one and scan chains 2,4 to scan input two scan. Therefore, chains 1,3 and 2,4 have the same data. Since the vector can be loaded using configuration zero for the first two shift cycles and the configuration one for the last two shift cycles, ATPG post processing will select the configurations {0,0,1,1} for this test vector.

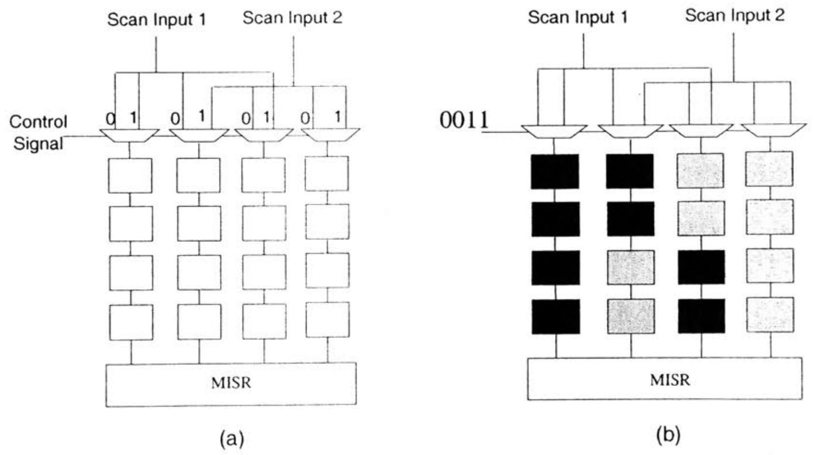


Figure 3-6: *Selecting the configurations for each shift cycle in dynamic testing.*

In prime based reconfiguration, the number of configurations that are available is limited. Similarly, in this example only two configurations are available. Therefore, it is possible to have test vectors that cannot be applied in any of the available configurations. Consider the circuit in Figure 3-6(a) again and assume that the following ATPG test vector needs to be loaded into the scan chains.

Scan chain 1: {1,1,1,1}
Scan chain 2: {1,1,0,0}
Scan chain 3: {0,0,1,1}
Scan chain 4: {0,0,1,1}

In the last two shift cycles of this vector, scan chains 1,3,4 require a value of one and scan chain 2 requires a value of zero. This vector cannot be loaded using either of two available configurations, since scan chains 1,2 must share the same data when the control signal is zero and scan chains 2,4 must share the same data when the control signal is one. If there is no configuration that can load data required in a given shift cycle, the configuration that creates the minimum number of incorrect test bits is selected. In this example, picking either configuration will load one incorrect bit to the scan chains. If a control signal of zero is used chain 1 or 2 will have one incorrect bit and if control signal one is used chain 2 or 4 will have one incorrect bit. In practice, dynamic configurations are used only on a small set of faults with a relatively large set of configurations to select from. Experimental results[9] show that the required values can be loaded using dynamic configurations under these circumstances.

[9] Given in Section 4.2

3.3.2 Creating the Configurations

Prime based reconfiguration uses an iterative process to create new configurations and generate test data until the required fault coverage is achieved. Let M denote the number of scan inputs used during a given configuration. Since a majority of the scan cells that are dependent on each other are either in the same scan chain or in adjacent scan chains, configurations are obtained by assigning every M^{th} scan chain to the same scan input. Figure 3-5 shows the configurations for M=2 and M=3 in a design with six scan chains. We start with a value of M=2 for the first configuration and keep increasing the value of M to determine the undetected faults. The following steps describe the process of assigning scan chains to the scan inputs.

1. Start with M=2.
2. Assign the scan chains to the scan inputs with every M^{th} scan chain being connected to the same scan input. N scan chains are connected to M scan inputs with each input driving N/M scan chains.
3. After the configuration is created, ATPG is run and the remaining undetected faults are determined. We will call this list of undetected faults F.
4. Once ATPG has been run on this configuration, we check whether the number of remaining undetected faults (F) is below a certain threshold value set by the DFT engineer. If the size of F is below that threshold, we go to step 5. Otherwise we move onto step 7.

5. ATPG is rerun with the assumption that the configuration can be changed at every shift cycle, using dynamic configurations. The configuration for each shift cycle is selected using ATPG post-processing.
6. If all the faults in F are not detected using the dynamic configurations the faults detected using dynamic configurations are ignored and the undetected faults list is set back to F.
7. The value of M is now increased to the next prime number and the process is repeated from step 2 until one of the following stopping criteria is met.
   (a) The maximum number of scan inputs has been used. The value of M cannot be increased because the designer cannot afford to use any more scan input pins.
   (b) All the faults in the circuit have been detected.

If the process is stopped because of stopping criterion (a) the results from step 5 are used to detect as many of the remaining faults as possible.

3.3.3 The Advantage of Using Prime Numbers

Prime based reconfiguration uses prime numbers to determine the scan inputs that are connected to each scan chain. In this section, we justify our use of prime numbers through a detailed explanation of the intuition that led to its selection, without attempting a formal proof. When creating multiple configurations, we would like to minimize the number of scan inputs and the number of configurations needed. The number of scan inputs should be minimized because an increase in the number of inputs reduces the parallelism and increases the data volume. The number of configurations should be minimized because the area overhead created by the control logic increases with the number of configurations created. If *M* were set to be incremented by one in Step 7 above, *M* would take all the integer values until the stopping criteria is met and this would create a large number of configurations. Using prime numbers for *M* instead of all integers reduces the number of configurations needed and does not reduce the number of faults that are detected.

It can be shown that using all integer values for *M* is wasteful. Figure 3-7 shows two configurations with 8 scan chains. In the first configuration (Figure 3-7(a)), where *M*=2, every other scan chain is connected to the same input and in the second configuration (Figure 3-7(b)), where *M*=4, every fourth scan chain is connected to the same input. In the first configuration, scan chains 1,3,5,7 must have the same values and in the second configuration scan chains 1,5 and 3,7 must have the same values loaded. Therefore, only half of the dependencies of the *M*=2 configuration are broken in the M=4 configuration.

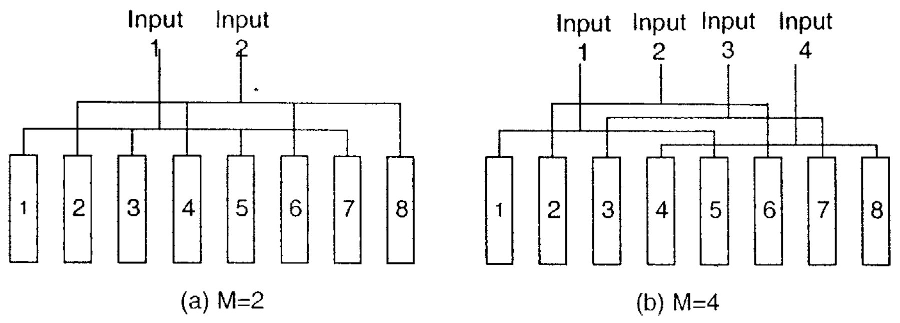
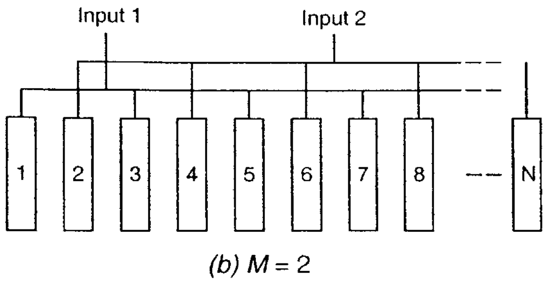


Figure 3-7: *The dependencies created by configurations of M=2 and M=4. The M=4 configuration has half the dependencies created by the M=2 configuration.*

It is also important to note that we use the M=2 configuration instead of the M=4 configuration, even though the M=4 configuration has fever dependencies. We make this choice because the test data volume depends on the number of scan inputs used. As the number of scan inputs increases the test data volume increases proportionately, since each scan input has to be loaded with the test data corresponding to that input. Therefore, this synthesis method attempts to detect as many faults as possible using low values of M.

As mentioned before, a goal of this process is to detect the faults using as few configurations as possible. The dependencies introduced in a given configuration can prevent a fault from being detected in that configuration. To detect such an undetectable fault, we require another configuration that does not have the dependency that prevented the fault from being detected. Therefore, we should minimize the common dependencies between the configurations that we create.

The Least Common Multiple (LCM) of a set of numbers is the smallest common multiple of the numbers in the set. For example, the LCM of 2 and 3 is 6 and the LCM of 3, 4 and 6 is 12. If S_{ij} is the LCM of the number of inputs in two configurations, i and j, the common dependencies occur at every S_{ij}th scan chain. For example, in the configurations given in Figure 3-7, $S_{2,4}$ is four and the common dependencies occur in every fourth scan chain. Now consider the configurations of M=2 and M=3, given in Figure 3-9. In this example, the value of $S_{2,3}$ is six and the common dependencies occur at every sixth scan chain.



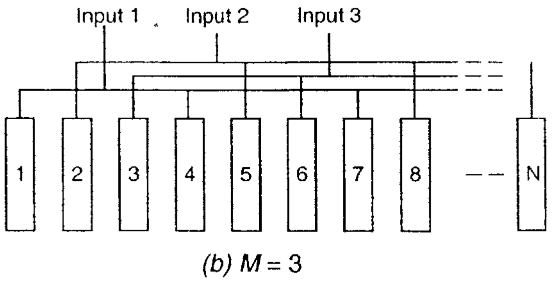


Figure 3-8: *The dependencies created by configurations of M=2 and M=3.*

When the value of S_{ij} increases, so does the interval between common dependencies, in terms of scan chains. As a result, the number of common dependencies in the two configurations decreases. Increasing the values of M for the two configurations will obviously increase the LCM. However, this will also increase the number of input pins. We would like to use a minimum number of input pins, because they are limited in quantity and also because using a large number of input pins increases the test data volume. Therefore, configurations that have a large value of S_{ij} with a relatively small number of inputs are ideal.

Consider two configurations $M=M_1$ and $M=M_2$. If M_1 and M_2 have a common factor, the LCM of M_1 and M_2 is only a fraction of M_1M_2. Let the common factor be C. We call the common factors *wasted inputs*, since we can obtain the same LCM by using the configurations M_1/C and M_2 or M_1 and M_2/C. Therefore, we would like to use a set of configurations that do not have any common factors. Since we want to start detecting faults in configurations with a low value of M, an increasing sequence with no common factors would be ideal. The prime number sequence, which is an increasing sequence with no common factors, satisfies this condition. Therefore, we use prime numbers sequence to create multiple configurations in this synthesis method.

It is important to note that the prime number sequence has not been proved to be the most effective sequence for creating configurations and other more effective sequences might exist. However, since the prime number sequence provides us with a set of configurations that provides satisfactory results, it was considered an adequate solution.

3.4 Optimizing the Mapping Logic

In both compatibility based and prime based reconfiguration, the mapping logic of RSSA that connects the scan inputs to the scan chains can be optimized after creating the configurations. In the definition of RSSA, in Section 2.2, a $2M$-way multiplexer is placed in front of each scan chain. Figure 2-4 from Section 2.2 is repeated below for easy reference.

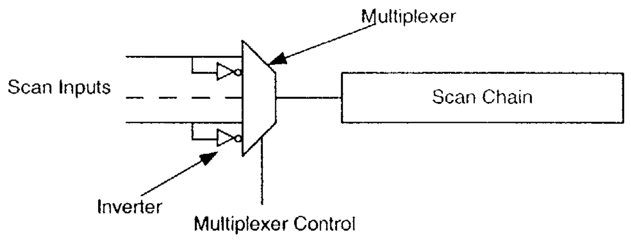


***Figure 3-9**: The old mapping logic. The multiplexer control signal can select one of all possible inputs to the scan chain.*

The scan input that drives each scan chain is selected using the control signal of the corresponding multiplexer. Since the scan chains are driven by different scan inputs in a given configuration, each multiplexer requires an independent control signal. For example, one scan chain might be driven by the first scan input and another scan chain might be driven by the inverse of the second scan input. The control signals of the multiplexers that drive these two scan chains clearly need to be different. Each control signal must be log($2M$) bits wide because $2M$-way multiplexers are used. Assume there are N scan chains in the circuit. Creating an independent control signal for each multiplexer requires a total of N control signals that have to be loaded through primary inputs. Since each control signal is log(2*M*) bits wide, a total of *N*[log(2*M*)] control wires are needed. If a large number of scan chains are used, the need for independent control signals introduces a large and impractical hardware overhead, because *N*[log(2*M*)] input pins are needed to apply the control signals. Since our goal is to indeed create a large number of scan chains and load them in parallel, this mapping logic needs to be optimized.

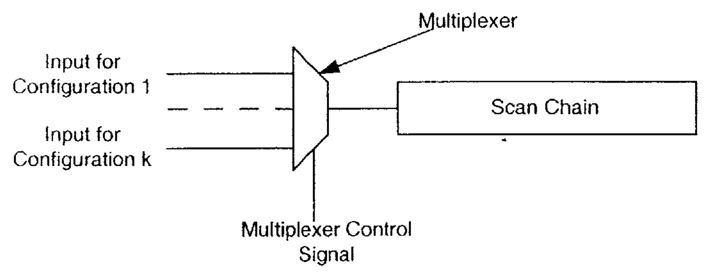

Multiplexer
Input for Configuration 1
Scan Chain
Input for Configuration k
Multiplexer Control Signal

Figure 3-10: *The optimized mapping logic. A multiplexer control signal of i picks the scan input that is needed in the* i^{th} *configuration.*

The following method is used to optimize the mapping logic after the configurations are determined. Let *k* be the number of configurations that are needed. Instead of using a 2*M*-way multiplexer that can select between all possible input configurations, we use a *k*-way multiplexer that can select an input based on the configuration being used. The new mapping logic is shown in Figure 3-10. The inputs to the multiplexer are rearranged such that the first input is the input used in configuration one and the k^{th} input is the input used in configuration *k*. If the same input is used in multiple configurations, multiple connections are made from the input to the multiplexer. For example, if the scan input *i* is used in both configurations one and three, scan input *i* is connected as the first and third inputs of the multiplexer. Under this new mapping logic, a multiplexer control signal of some value j will select the scan input needed for configuration j in all the multiplexers. Therefore, we can share the same control signal for all the multiplexers and avoid all the additional wiring that is needed with the original mapping logic.

Chapter 4

Experimental Results

This chapter evaluates the performance of the proposed reconfigurable shared scan-in architecture. Compatibility based reconfiguration and prime based reconfigurations, the two synthesis procedures for RSSA are analyzed separately and presented in Sections 4.1 and 4.2. A comparison of the two procedures is presented in Section 4.3. The analysis is performed using four of the larger ISCAS'89 benchmark circuits [38] and three much larger proprietary circuits. Table 4-1 shows some characteristics of the test circuits. The existing scan chains of the designs are removed and new scan chains are created as needed in each experiment. The faults are determined using the stuck-at fault model. All the flip-flops in the circuits are used as scan cells, because the designs are all full scan. The maximum cone size gives the size of the largest cone with respect to the number of gates.

Circuit	#Gates	#Faults	# Flip-Flops	Max. Cone Size (gates)	Source
Circuit A	230k	481k	9700	1887	Synopsys
Circuit B	390k	554k	12500	916	Synopsys
Circuit C	1083k	2740k	69000	5454	Synopsys
S13207	2573	20k	669	468	ISCAS '89
S15850	3448	22k	597	419	ISCAS '89
S38417	8709	50k	1636	302	ISCAS '89
S38584	11448	66k	1426	346	ISCAS '89

Table 4-1: Characteristics of the test circuits

Each circuit was run through each of the proposed schemes and compared against the results of regular scan based ATPG and Illinois Scan as described in [8]. All the experiments were run using the Synopsys tools DFT compiler™ and TetraMAX® [39].

In scan based testing, the test data is loaded into the scan chains and then the circuit is clocked in functional mode to observe the response of the circuit. When the circuit is clocked in functional mode, additional data is loaded through the primary input pins of the circuit. If x is the number of primary inputs of the circuit, then x bits of data needs to be loaded during each test vector. Since loading primary inputs is a serial process, the test data volume reduction given by the parallel loading of scan chains is scaled down[10]. Therefore, we create a shift register called a *wrapper chain* around the primary inputs of the circuit, such that this data can be loaded in parallel as well. The wrapper chain connects all the primary inputs of the circuit into one long chain that can be loaded using one input .The wrapper chain is then treated as a scan chain and the scan inputs are used to load data into the wrapper chain. If the wrapper chain is longer than the longest scan chain, multiple wrapper chains are created such that none of them are longer than the longest scan chain. This is important because the test application time is dependent on the length of the longest scan chain.

[10] The reduction can be calculated using Amdahl's Law. See Chapter 2.

4.1 Compatibility Based Reconfiguration

The experimental results of running a set of test circuits on RSSA with compatibility based reconfiguration, is presented in this section. The results are compared against the results of regular scan based ATPG and Illinois Scan. We will first define the Data Volume Reduction (DVR) for Illinois Scan and RSSA with compatibility based reconfiguration.

We introduce the following definitions:

P	–	Number of test patterns required for regular ATPG
P_b	–	Number of broadcast mode test patterns in Illinois Scan
P_s	–	Number of serial mode test patterns in Illinois Scan
N	–	The number of internal scan chains
M_i	–	The number of scan inputs in Illinois Scan
P_c	–	Total number of test patterns required by all configurations of RSSA with compatibility based reconfiguration
M_c	–	The number of scan inputs in RSSA with compatibility based reconfiguration

During the broadcast mode of Illinois Scan, N scan chains are loaded using M_i inputs. The number of bits loaded per test pattern is a factor of N/M_i lower than in regular scan testing and the serial mode of Illinois Scan. Therefore, the DVR of Illinois Scan compared to regular scan testing is:

$$DVR_{IS} = P/(P_b(M_i/N) + P_s)$$

Similarly, in the RSSA with compatibility based reconfiguration N scan chains are loaded using M_c scan inputs. The test data loaded per test pattern is a factor of N/M_p lower than in regular scan testing. Therefore, the DVR of RSSA with compatibility based reconfiguration compared to regular scan testing is:

$$DVR_{RSSA} = P / P_c(M_c/N)$$

Circuit	N	M_c	L	K	Regular ATPG Patterns	Illinois scan			Proposed synthesis procedure				
						Broad. Pat.	Serial Pat.	DVR	Patterns	Conf.	DVR	ΔFC #faults	Percentage ATPG overhead
S13207	42	2	21	212	145	177	45	2.71	243	3	12.53	0	261
	80	2	11		148	184	58	2.36	296	5	20.00	0	802
S15850	42	2	20	183	104	213	32	2.47	319	5	6.85	0	833
	77	2	10		100	184	46	1.97	448	12	8.59	0	5689
S38417	67	2	27	99	139	405	26	3.65	491	4	9.48	0	392
	129	2	14		136	542	40	2.81	781	5	11.23	0	1816
S38584	74	2	25	147	232	322	59	3.43	477	5	18.00	0	686
	139	2	13		229	348	93	2.34	636	8	25.02	0	2951
Circuit A	130	2	77	432	977	1474	117	6.99	1545	3	41.10	32	152
	252	2	39		989	1809	232	4.01	2205	3	56.51	39	1101
	487	4	20		970	1865	207	4.36	2264	4	52.16	0	1029
	487	2	20		970	2137	249	3.76	3169	4	74.53	141	4323
Circuit B	135	2	100	282	711	1293	82	7.03	1455	3	32.98	0	114
	263	2	51		754	1652	161	4.34	2011	4	49.30	-194	605
	516	4	26		747	1599	206	3.42	2028	4	47.52	-361	987
	516	2	26		747	2055	228	3.17	2328	4	82.79	-808	3345
Circuit C	135	2	583	264	2227	2441	56	24.16	2500	2	60.13	46	7
	269	2	277		2325	2636	129	15.65	2773	2	112.77	23	25
	537	4	135		2330	2665	350	6.30	3038	2	102.96	25	50
	537	2	135		2230	3061	405	5.36	3517	2	170.25	0	239

***Table 4-2**: Experimental results for test data volume reduction*

The experimental results for compatibility based reconfiguration are given in Table 4-2. In each experiment we use the same number of scan chains and scan inputs for both Illinois Scan and RSSA. Therefore, $M_i = M_c$ and this value is denoted as M_c in Table 4-2. N is the number of scan chains and includes the wrapper chains that are created around the primary inputs. L is the length of the longest scan chain. If F is the number of flip-flops in the circuit and W is the number of wrapper chains in the circuit,

$$L = \lceil F/(N-W) \rceil$$

K is the size of the largest cone of logic in terms of the number of scan cells in the cone. The last two columns in Table 4-2 show the deviation in fault coverage and percentage ATPG runtime overhead of compatibility based reconfiguration with respect to regular ATPG.

A test vector is applied by loading data into N scan chains using M_c scan inputs. Therefore, only M_cL bits of data are specified using the ATPG engine. Detecting a given fault requires test data to be loaded into, at most all the scan cells that drive the fault being tested. This is explained in Section 3.1. Since K is the size of the largest cone, no more than K scan cells need to be loaded to test any given fault. If K is larger than M_cL, the scan cells of the largest cone must be spread over more than M_c scan chains. Therefore, two or more scan chains that contain scan cells from the largest cone must be loaded using the scan input. Assume that f is a stuck-at fault in the largest cone and that x, y are two scan cells that need to be specified to test f. The fault f is undetectable if all of the following criteria are satisfied.

1. The values needed in x and y are not compatible (logic-0 and logic-1)
2. x and y are in the same shift position relative to the scan input.
3. x and y share the same scan input

If all three of these conditions are satisfied, the two scan cells must share the same data, but require conflicting values to be loaded. This makes fault f undetectable.

In most cases, only a fraction of the K scan cells are needed to detect a fault and these scan cells can be in M_c or fewer scan chains. However, requiring less than M_cL scan cells to detect a fault does not guarantee that the scan cells are in M_c or fewer scan chains. The scan chain creation algorithm, described in Section 3.1, tries to group scan cells from the same cone together, but might not be able to do so when many different cones share the scan cells. The algorithm might place scan chains from the same cone of logic in many scan chains even if the cone size is smaller than L. Therefore, the scan cells needed to detect a fault could be in more than M_c scan chains even if K is smaller than M_cL.

When we increase the parallelism by increasing *N*, the length of the scan chains (*L*) gets shorter. Since *L* is decreasing and *K* does not change, the scan cells from a given cone are spread out over more scan chains. Now the scan cells that need to be specified to detect a given fault will also be spread over more scan chains. This increases the chances of a fault being undetectable in RSSA. An example of such undetectable faults can be seen in the last three experiments with Circuit B.

Increasing the parallelism also decreases the number of configurations in which a given fault can be detected. In the above discussion, we talked about the extreme case, where a given fault cannot be detected in any configuration. When the number of configurations in which a fault can be detected decreases, the compatibility analysis engine in ATPG has to do more work to search for configurations. This increases the runtime of ATPG significantly, especially when undetectable faults cause the analysis engine to search through all the configurations. Therefore, the ATPG runtime increases as the parallelism increases. This can be seen clearly in the last column of Table 4-2, which shows that the ATPG runtime increases with *N*. For example, in S15850 the percentage ATPG overhead increases from 833% to 5689% as the number of scan chains is increased from 42 to 77.

Figure 4-1 shows how the DVR of compatibility based reconfiguration compares with that of Illinois Scan. Each pair of bars represents an experiment from Table 4-2. The ISCAS circuits have two experiments each and the proprietary circuits have four experiments each. The numbers following each circuit name denote the number of scan chains and the number of scan inputs used in the experiment. For example, S13207-42-2 represents testing the circuit S13207

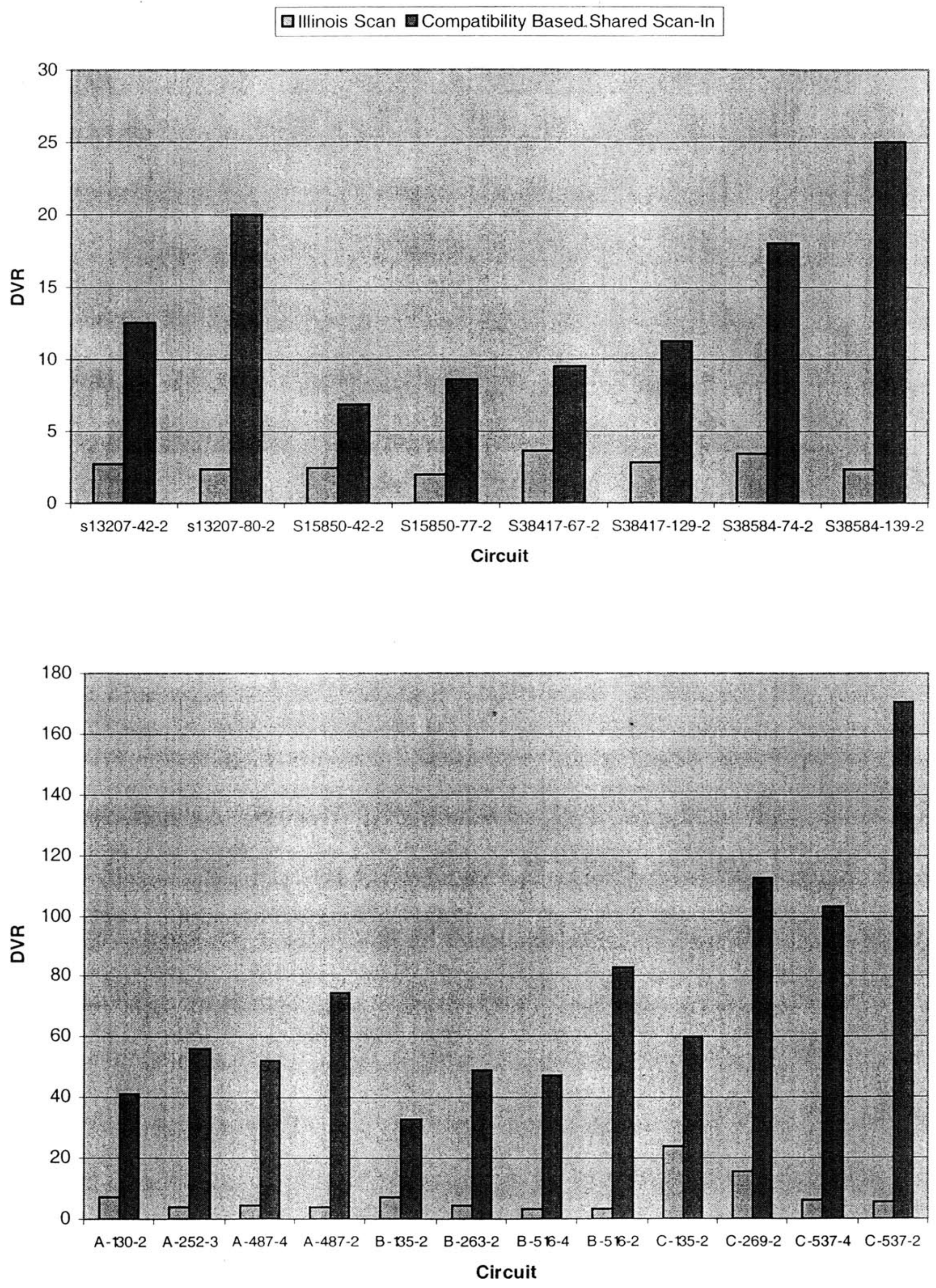


Figure 4-1: *The DVR for Illinois Scan and the proposed compatibility based reconfiguration. The numbers after the circuit name denote the number of scan chains (N) and scan inputs (M).*

with 42 scan chains and 2 scan inputs and S13207-80-2 represents testing the circuit S13207 with 80 scan chains and 2 scan inputs. The experimental results for each circuit show that the DVR of RSSA increases as the number of scan chains loaded in parallel increases, while the DVR of Illinois Scan decreases as the parallelism increases. This can be most clearly seen in the results for Circuit C. Table 4-2 shows that the number of serial mode patterns used in Illinois Scan increases as the parallelism is increased, which leads to the reduction in DVR. When the parallelism is increased, the number of test patterns that are required to test the circuit increases in RSSA as well. However, since these additional vectors are also broadcast mode vectors, the overhead is minimal. The DVR continues to increase since the gains achieved by the added parallelism are much greater than the overhead of the extra test vectors. Since RSSA is able to detect faults with a much higher parallelism than that of Illinois Scan, the DVR of RSSA is significantly larger for all circuits tested.

In general, as full scan circuits get larger, the number of scan cells in the circuits increase proportionately. When the number of scan cells increases, the length of each scan chain also increases, for a given number of scan chains. However, as the circuit size increases, the largest cone sizes (*K*) remains relatively constant, as seen in Table 4-2. For example, let us compare the two circuits S13207 and proprietary Circuit C. The value of K is relatively similar in both circuits with a value of 212 for S13207 and 264 for Circuit C. However, the number of scan cells in Circuit C is two orders of magnitude larger than the number of scan cells in S13207. If the number of scan chains is kept constant, the length of each scan chain increases as the circuits get larger. Since the cone size stays relatively constant, it is more likely that the scan cells of a given cone will be in a small number of scan chains. The limiting factor for increasing parallelism is a large number of scan cells from the same cone being in many scan chains, as described earlier in this section. Therefore, we can increase the parallelism by a larger factor in the bigger circuits. The experimental results also show that, in general, RSSA performs better as the circuit size gets larger. All three proprietary circuits achieved much larger reductions in DVR than the smaller ISCAS circuits.

Let us now consider the Test Application Time Reduction (TATR) of RSSA with compatibility based reconfiguration. Here, we define:

P	–	Number of test patterns required for regular ATPG
P_c	–	Total number of test patterns required by all configurations of RSSA with compatibility based reconfiguration
N	–	The number of internal scan chains
M	–	The number of scan inputs in regular scan
M_c	–	The number of scan inputs in RSSA with compatibility based reconfiguration

The length of the scan chains in RSSA is a factor *N/M* lower that in regular scan testing since the number of scan chains is increased from *M* to *N*. Therefore, the TATR for RSSA compared to regular scan is:

$$TATR_{RSSA} = P / P_c (M / N)$$

Previously, we defined the DVR for RSSA compared to regular scan testing to be:

$$DVR_{RSSA} = P / P_c (M_c / N)$$

Therefore, we can obtain the following relation between the DVR and TATR:

$$TATR = (M / M_c) * DVR$$

This means that the DVR and TATR for RSSA are equal when the number of scan inputs used in regular scan and RSSA are the same. However, if more scan inputs are used in regular scan the TATR decreases. For example, assume that Circuit A had eight scan inputs in regular ATPG mode and two inputs with 32 scan chains in the proposed scheme. In this case,

$$\begin{aligned} \text{TATR} &= (M / M_c) * \text{DVR} \\ &= \tfrac{1}{4}\,\text{DVR} \end{aligned}$$

The TATR depends on the length of the scan chains in RSSA. To increase the TATR we would need to reduce the length of the scan chains. Consider a circuit that is being tested with the highest parallelism that allows for a maximum number of faults to be detected. The parallelism is defined, as the ratio N/M_C. We cannot reduce the length of the scan chains by simply increasing the number of scan chains, because this increases the parallelism. However, increasing both the number of scan chains and the number of scan inputs by the same ratio is acceptable, because this does not change the parallelism. Assume that we double both the number of scan chains and the number of scan inputs. The scan cells from each cone are now spread over twice as many scan chains. However, the fault coverage is not affected, because we now have twice as many inputs to load data into scan chains.

The TATR that can be achieved by using more input pins is limited by a few factors. A circuit only has a limited number of input pins and increasing the number of scan inputs is an additional test cost. The ATE cost increases with the number of pins on the tester. Therefore, testing chips with a large number of inputs pins is more expensive. Furthermore, as the scan chains become shorter the DFT synthesis becomes more constrained, because routing a large number of very short chains introduces a large overhead. This could create routing problems, especially if the scan cells in each scan chain are not close to each other in the topology of the circuit. Therefore, the test engineer must decide on how many scan chains and scan inputs should be used based on the characteristics of the circuit under test.

4.2 Prime Based Reconfiguration

The experimental results of RSSA with prime based reconfiguration are analyzed with the same circuits used to analyze compatibility based reconfiguration and the results are reported in this section. In prime based reconfiguration, the configurations are selected as described in Section 3.3, with the last configuration being a dynamic configuration. We will first define the data volume reduction (DVR) for RSSA with prime based reconfiguration.

Let M_p be the maximum number of scan inputs used in prime based reconfiguration. In this method, the scan chains are connected to M_i scan inputs in the i^{th} configuration, where $2<M_i<M_p$. If L_p is the length of the scan chains, M_i bits of data are loaded L_p times per test pattern in the i^{th} configuration. ATE's requires that any unused scan inputs be specified before loading the test vector. This requires one bit of data per unused pin. If V_i is the number of test patterns loaded in the i^{th} configuration, the test data volume (DV) for each static configuration is:

$$\begin{aligned} DV_{STATIC} &= \text{Patterns} * [(\text{Scan Pins Used} * \text{Scan Chain Length}) + \text{Unused Scan Pins}] \\ &= V_i\,[M_i L_p + (M_p - M_i)] \end{aligned}$$

In dynamic testing, all M_p scan pins are used to load data to the circuit. Since the scan chains length is L_p, M_pL_p bits of data need to be loaded per test pattern. If V_d is the number of test patterns loaded during the dynamic configuration, the test data volume for the dynamic configuration is:

$$\begin{aligned} DV_{DYNAMIC} &= \text{Dynamic patterns} * \text{Total Scan Pins} * \text{Scan Chain Length} \\ &= V_d M_p L_P \end{aligned}$$

The test data volume for the prime based reconfiguration is equal to the sum of the data volumes of each configuration, because the circuit is tested iteratively using all of these configurations. Therefore, the test data volume for all prime based configurations is:

$$DV_{PRIME} = DV_{STATIC1} + DV_{STATIC2} + \ldots + DV_{DYNAMIC}$$

During regular ATPG, each scan chain has its own scan input. Let M be the number of scan inputs. If L is the length of a scan chain, ML bits of data need to be loaded per test patter. If V test patterns are loaded during regular ATPG, the test data volume of regular ATPG is:

$$DV_{REG} = \text{Test Patterns} * \text{Scan Chains} * \text{Scan Chain Length}$$

$$= VML$$

The data volume reduction (DVR) for RSSA with prime based reconfiguration is related to the data volume of regular scan based ATPG and the data volume of RSSA with prime based reconfiguration as follows:

$$DVR_{PRIME} = DV_{REG} / DV_{PRIME}$$

Design	N	M	L	Regular ATPG	Illinois Scan			RSSA with Prime Based Reconfiguration		
				Patterns	Broad. Pat.	Serial Pat.	DVR	Patterns	M_p	DVR
Circuit A	487	2	20	970	2137	249	3.76	2910	7	64.42
Circuit B	516	2	26	747	2055	228	3.17	2767	7	48.00
Circuit C	537	2	135	2230	3061	405	5.36	3715	7	138.6
S13207	80	2	11	148	184	58	2.36	305	7	9.30
S15850	77	2	10	100	184	46	1.97	408	11	3.01
S38417	129	2	14	136	542	40	2.81	731	5	9.14
S38584	139	2	13	229	348	93	2.34	595	7	13.56

Table 4-3: *The data volume reduction of RSSA with prime based reconfiguration*

Table 4-3 gives the experimental results in data volume reduction for the prime based reconfiguration in RSSA. N is the number of internal scan chains used by Illinois Scan and prime based reconfiguration. M is the number of scan inputs used by regular ATPG and Illinois Scan. In regular ATPG the M scan inputs are connected to M scan chains, while in Illinois Scan the M scan inputs are connected to N scan chains. L is the length of the scan chains in Illinois Scan and the prime based reconfiguration. M_p is the maximum number of scan inputs used by prime based reconfiguration. If $M_p = 7$, prime based reconfiguration uses the configurations of $M = 2, 3, 5, 7$ and a dynamic configuration that switches between these four configurations, resulting in a total of five configurations.

In compatibility based reconfiguration, we were able to test all the circuits using two scan inputs. However, as Table 4-3 shows, a larger number of scan inputs are required when using prime based reconfiguration. The reason for this increase is the fact that prime based reconfiguration relies on increasing the input pins to break dependencies, unlike compatibility based reconfiguration that intelligently picks new configurations that can break dependencies. However, even using this less efficient variant of RSSA, we can still obtain a much higher DVR than Illinois Scan, as seen in Figure 4-2. Another common trend between the two synthesis methods is that the DVR increases with the circuit size. The reason is again because the number of scan cells increase with size, while the number of scan cells that need to be specified to test a fault stay relatively constant. A comparison of compatibility based reconfiguration and prime based reconfiguration is given in the next section.

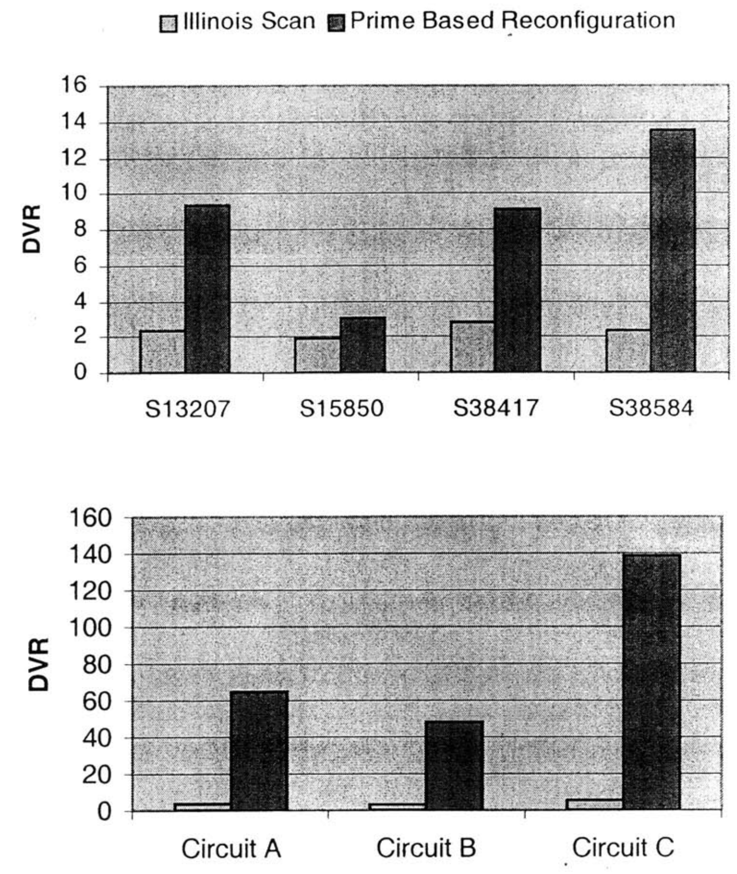


Figure 4-2: The DVR for Illinois Scan and the proposed prime based reconfiguration.

Design	N	M	L	Regular ATPG	Illinois Scan			RSSA with Prime Based Reconfiguration		
				Patterns	Broad. Pat.	Serial Pat.	DVR	Patterns	M_p	DVR
Circuit A	487	2	20	970	2137	249	3.76	2910	7	81.17
Circuit B	516	2	26	747	2055	228	3.17	2767	7	69.65
Circuit C	537	2	135	2230	3061	405	5.36	3715	7	161.17
S13207	80	2	11	148	184	58	2.36	305	7	19.41
S15850	77	2	10	100	184	46	1.97	408	11	9.44
S38417	129	2	14	136	542	40	2.81	731	5	12.00
S38584	139	2	13	229	348	93	2.34	595	7	26.75

***Table 4-4:** The test application time reduction of prime based reconfiguration in the proposed architecture*

The test application time reduction (TATR) depends on ratio of the length of the longest scan chain during regular ATPG and the prime based testing.

We introduce the following definitions:

P	– Number of test patterns required for regular ATPG
P_p	– Total number of test patterns required by all configurations of the prime based testing
N	– The number of internal scan chains
M	– The number of scan inputs in regular scan

The TATR for prime based reconfiguration compared to regular ATPG is:

$$TATR_{PRIME} = P / P_p (M / N)$$

Table 4-4 gives the experimental results in test application time reduction for the prime based reconfiguration method. The TATR of prime based configuration is larger than the DVR, because the test data volume increases as more input pins are used and the test application time is not affected. The test application time is not affected since the length of the scan chains does not change.

To determine the effectiveness of dynamic testing we tested each circuit using the regular prime based reconfiguration with and without dynamic configurations. When prime based reconfiguration is used without dynamic configurations, we call the configurations *static only* configurations. When using static only configurations the number of input pins (*M*) is increased until all the faults are detected. Table 4-5 shows a comparison of static only testing with the regular static and dynamic testing.

Design	Static Only			Static + Dynamic			
	Patterns	Configurations	Pins	Total Patterns	Dynamic Patterns	Configurations	Pins
Circuit A	2922	6	13	2910	22	4	7
Circuit B	2780	5	11	2767	34	4	7
Circuit C	3712	5	11	3715	25	4	7
S13207	305	5	11	305	6	4	7
S15850	409	7	17	408	12	5	11
S38417	735	6	13	731	16	3	5
S38584	562	6	13	595	45	4	7

Table 4-5: *Comparison of static only vs. and static and dynamic testing*

When dynamic configurations are not used, the circuit requires more input pins and more configurations for all the faults to be tested. For example in Circuit A, when dynamic configurations are used only four configurations and seven input pins are needed to test all the faults. However, when static only configurations are used, six configurations and thirteen input pins are needed. Therefore, three static configurations with six extra input pins are needed to detect the faults that were detected in one dynamic configuration. Since RSSA can handle dynamic configurations at no extra cost and input pins are expensive, the ability to use dynamic configurations provides a significant benefit in prime based reconfiguration.

4.3 Compatibility based vs. Prime based Reconfiguration

In this section, we compare the experimental results of compatibility based reconfiguration and prime based reconfiguration. Figure 4-3 provides a comparison of the data volume reduction (DVR) given by Illinois Scan, prime based reconfiguration and compatibility based reconfiguration of the test circuits. For Illinois Scan and compatibility based reconfiguration, we use the largest DVR for each circuit from Table 4-2. It is clear that both synthesis methods of RSSA are able to provide a much larger DVR than Illinois Scan. Of the two synthesis methods, the compatibility based method is able to provide a higher DVR for all tested circuits.

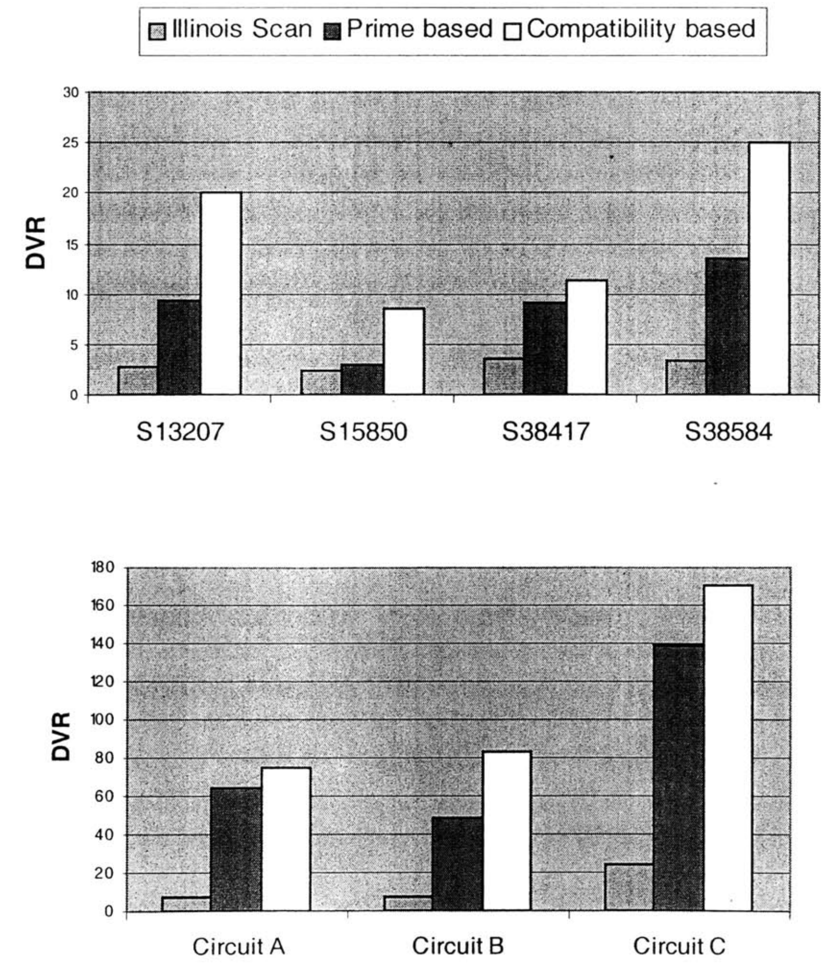


Figure 4-3: Comparison of Illinois Scan, Prime based and Compatibility based testing.

The compatibility based reconfiguration technique incurs a large ATPG overhead because it uses computationally intensive compatibility analysis. This analysis allows us to test a circuit with a large number of internal scan chains and a few scan inputs, but has a high ATPG runtime cost when the analysis engine is pushed to its limit, with a large number of chains being loaded in parallel[11]. Although ATPG runtime is a one-time cost, the test generation time can be critical in certain design cycles. The prime based reconfiguration technique does not use any extra analysis and therefore has no ATPG overhead. However, this scheme requires more input pins than the compatibility based scheme, because it relies on a larger number of scan inputs to eliminate conflicts. Table 4-6 summarizes the differences of Illinois Scan, prime based reconfiguration and compatibility based reconfiguration.

	Illinois Scan	Prime based	Compatibility based
DVR	Low	High	Very high
ATPG overhead	No	No	High
#Input Pins	Low	High	Low

***Table 4-6:** Comparing Illinois Scan, prime based reconfiguration and compatibility based reconfiguration*

Both proposed schemes provide a high DVR with each scheme having a different overhead over regular scan testing. This allows the DFT engineer in charge of testing a circuit to decide whether it is more important to have a higher DVR while incurring a high ATPG overhead or tradeoff some DVR for no ATPG overhead and a small number of extra input pins.

[11] See Table 4-2.

Chapter 5

Conclusions

The cost of testing an integrated circuit has become an increasing concern for test engineers as the cost of testing increases relative to the cost of manufacturing. The decreasing size and cost of circuit components has led to an increase in the average circuit size. As the circuit size increases, both the test data volume and test application time of a circuit increase. Circuits with a large test data volume require expensive testers that can handle a large amount of data. Furthermore, as the test application time increases, more testers may be needed to test a given set of chips within the required time.

Much research has been done in the area of reducing the cost of integrated circuit testing, but the solutions presented so far either give relatively small benefits or come with a large hardware overhead. One of these solutions, Illinois Scan, which loads the scan chains in parallel, has potential, but is limited by the dependencies created when loading many chains in parallel. In this thesis, we presented a new test architecture (RSSA) for loading the scan chains in parallel, that removes these dependencies by reconfiguring the inputs that drive each scan chain during the testing process. The mapping logic of RSSA consists of a multiplexer per scan chain and therefore comes with a low hardware overhead.

We proposed two methods for selecting the scan inputs that drive each scan chain. The first solution, called compatibility based reconfiguration, uses an analysis engine to determine the scan input that should drive each scan chain. This engine uses ATPG to divide the scan chains into compatibility classes, with all the scan chains in a compatibility class being driven by the same input. Experimental results show that this method provides large reductions in both test application time and test data volume. The drawback of this method is that the ATPG based compatibility analysis can be very time consuming. It should be noted that this is a one-time cost during the design cycle of the chip. However, this might be a problem for some design engineers working with a tight deadline to start manufacturing.

To eliminate the time overhead of compatibility analysis, we presented a second solution that breaks dependencies by simply using more scan inputs. The test architecture's ability to change configurations during the application of a test vector (dynamic configurations) is used to minimize the number of scan inputs needed. The data volume reduction of this method is lower than that of compatibility based reconfiguration. However, this method is still much more effective than the standard parallel loading architecture, Illinois Scan.

The two solutions provide test engineers an opportunity to select the option that best fits the needs of the circuit under test. In most cases, we expect compatibility based reconfiguration to be used because of the higher reduction in test cost. However, in some cases, due to tight delivery schedules, the less efficient prime based reconfiguration might be used to reduce the ATPG runtime.

Further research in the following areas will help improve the effectiveness of RSSA. When creating the scan chains, the number of inter scan chain dependencies can be reduced if we were able to order the scan cells within a chain for that purpose. However, the placement tool needs the flexibility to order the scan chains to satisfy routing constraints. If the DFT and placement tools could work together to create the scan chains, we would be able to create an ordering that satisfies the routing constraints and minimizes dependencies.

We mentioned above that the runtime of compatibility analysis was a limiting factor in using compatibility based reconfiguration. Much work needs to be done in researching new and faster methods of creating compatibility classes or alternative methods that can break the dependencies.

Finally, we discussed that current ATPG technology cannot handle multiple constraints when creating test vectors. Only one set of constraints can be specified before test generation. This reduces the effectiveness of dynamic configurations because ATPG cannot fully utilize the architecture's ability to use a different configuration during each shift cycle. We will not be able to take full advantage of dynamic configurations until advances in ATPG technology allow for test generation with multiple constraints.

As mentioned above, there are many improvements that can be made to make this reconfigurable shared scan-in architecture more effective. However, we have shown that RSSA provides large gains relative to other scan based test methods, even in its current state. In addition, this architecture does not introduce a large hardware overhead. Furthermore, we showed that the architecture works better when testing large circuits. This is important for the future applicability of RSSA since we expect circuit sizes to continue increasing. In conclusion, we anticipate that test strategies based on this architecture will be used in industry as an alternative to BIST based test solutions.

Bibliography

[1] International Technology Roadmap for Semiconductors, 1999 edition.

[2] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing.* Kluwer Academic Publishers 2002.

[3] K. M. Butler and M. R. Mercer, "Quantifying Non-Target Defect Detection by Target Fault Test Sets," 1991 European Test Conf., Apr. 1991

[4] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design.* IEEE Computer Society Press, New York, 1990.

[5] M. J. Y. Williams and J. B. Angell, "Enhancing testability of large-scale integrated circuits via test points and additional logic." IEEE transactions on Computing vol. C-22 pp. 46-60, Jan. 1973

[6] E. B. Eichelberger and T. W. Williams, 'A logic design structure for LSI testability, " Journal of Design Automation and Fault Tolerant Computing vol. 2, pp. 165-178, May 1978.

[7] V. G. Oklobdzija and M. D. Ercegovac, "Testability enhancement of VLSI using circuit structures." Proc. IEEE ICCC, pp. 198-201, 1982

[8] I. Hamzaoglu and J. H. Patel, "Reducing Test Application Time for Full Scan Embedded Cores," Proc. IEEE FTCS, 1999, pp.260-267.

[9] A. Pandey and J. H. Patel, "Reconfiguration Technique for Reducing Test Time and Test Data Volume in Illinois Scan Architecture Based Designs," Proc. IEEE VTS, 2002, pp.9-15.

[10] S. Samaranayake, N. Sitchinava, R. Kapur, M. Amin and T. W. Williams, "Dynamic Scan: Driving Down the Cost of Test," IEEE Computer, October 2002, pp.65-70.

[11] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, B. Keller and B. Koenemann, "OPMISR: The Foundation for Compressed ATPG Vectors," Proc. ITC, 2001, pp. 748-757.

[12] J. Rajski, J. Tyszer, M. Kassab, N. Mukerjee, R. Thompson, K. H. Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide and J. Qian, "Embedded Deterministic Test for Low Cost Manufacturing Test," Proc. ITC, 2002, pp. 301-310.

[13] M. J. Geuzebroek, J. Th. van der Linden and A. J. van de Goor, "Test Point Insertion that facilitiates ATPG in reducing test time and data volume," Proceedings of the International Test Conference, 2002, pp. 138-147.

[14] A. Jas and N. A. Touba, "Test Vector Decompression via Cyclical Scan Chains and Its Application to Testing Core-Based Designs," Proceedings of the International Test Conference, 1998, pp. 458-464.

[15] A. Jas, B. Pouya and N. A. Touba, "Virtual Scan Chains: A means for reducing scan length in cores," Proceedings of the VLSI Test Symposium, 2000, pp. 73-78.

[16] K. D. Wagner, "Robust Scan-Based Logic Test in VDSM Technologies," IEEE Computer, November 1999, pp. 66-74

[17] K. J. Lee, J. J. Chen and C. H. Huang, "Using a single input to support multiple scan chains, " Dig Tech. Papers, IEEE/ACM Int. Conf. Computer-Aided Design, 1998, pp. 74-78.

[18] S. Y. Lee and K. K. Saluja, "An Algorithm to Reduce Test Application Time in Full Scan Designs," Proc. of the Int. Conf. on Computer Aided Design, 1992, pp. 17-20.

[19] D. K. Pradhan and J. Saxena, "A Design for Testability Scheme to Reduce Test Application Time in Circuits with Full Scan," Proc of the IEEE VLSI Test Symp., 1992, pp.55-60.

[20] E. M. Rudnick and J. H. Patel, “A Genetic Approach to Test Application Time Reduction for Full Scan and Partial Scan Designs,” Proc. of the Int. Conf. on VLSI Design, 1995, pp. 288-293.

[21] S. Lee and K. G. Shin, “Design for Test Using Partial Parallel Scan,” IEEE Trans. On Computer-Aided Design, 1990, vol. 9, pp. 203-211.

[22] S. Narayanan, R. Gupta and M. Breuer, “Optimal Configuring of Multiple Scan Chains,” IEEE Trans. On Computer-Aided Design, 1993, vol. 42, no. 9, pp. 1121-1131.

[23] S. Narayanan and M. A. Breuer, “Reconfiguration techniques for a single scan chain,” IEEE Trans. Computer-Aided Design, 1995, vol.14, no.6, pp. 750-765.

[24] B. Koenemann et al, “A SmartBIST Variant with Guaranteed Encoding,” Proc. ATS, 2001, pp. 325-330.

[25] P. Wohl, J. A. Waicukauski, T. W. Williams, “Design of Compactors for Signature-Analyzers in Built-In Self-Test,” Proc. of the IEEE ITC, pp. 54-63, 2001.

[26] D. Das and N. A. Touba, “Reducing Test Data Volume using External/LBIST Hybrid Test Patterns,” Proc. of the IEEE ITC, 2000, pp. 115-122.

[27] A. Jas, C. V. Krishna, and N. A. Touba, “Hybrid BIST Based on Weighted Pseudo-Random Testing: A new resource partitioning scheme,” Proc. of the IEEE VLSI Test Symp., 2001, pp. 2-8.

[28] R. Dorsch and H. J. Wunderlich, “Accumulator based deterministic BIST,” Proc. of the IEEE ITC, 1998, pp. 412-421.

[29] G.Hetherington, T.Fryars, N.Tamarapalli, M.Kassab, A.Hassan and J.Rajski, “Logic BIST for large industrial designs,” Proc. of the IEEE ITC, 1999, pp. 358-367.

[30] G. Kiefer, H. Vranken, E. J. Marinissen and H. J. Wunderlich, “Applications of deterministic logic BIST on industrial circuits,” Proc of IEEE ITC, 2000, pp. 105-114.

[31] A. Chandra and K. Chakrabarty, "Frequency Directed run length(FDR) codes with application to system-on-a-chip test data compression," Proc. of the IEEE VLSI Test Symp., 2001, pp. 42-47.

[32] S. Makar, "A Layout – based Approach for Ordering Scan Chain Flip-Flops," Proc. of the IEEE ITC, 1998, pp.341-347.

[33] C. Chen and S. K. Gupta, "A Methodology to Design Efficient BIST Test Pattern Generators," Proc. IEEE ITC, 1995, pp.814-823.

[34] I. Hamzaoglu and J. H. Patel, "Reducing Test Application Time for Built-in Self-Test Test Pattern Generators," Proc. IEEE VTS, 2000, pp.369-375.

[35] K. Chakrabarty, B. Murray, J. Liu and M. Zhu, "Test Width Compression for Built-in Self Testing," Proc. IEEE ITC, 1997, pp.328-337.

[36] E. Gizdarski and H. Fujiwara, "Fault Set Partition for Efficient Width Compression,"

[37] I. Bayraktaroglu and A. Orailoglu, "Test Volume and Application Time Reduction Through Scan Chain Concealment," Proc. ACM/IEEE Design Automation Conf., 2001, pp.151-155.

[38] ISCAS '89 Benchmark Information, http://www.cbl.ncsu.edu/CBL_Docs/iscas89.html

[39] Synopsys Test Tools, http://www.synopsys.com/products/test/test.html

4470-63