Hybrid Selective X-masking and X-canceling Multiple Input Signature Register for Test Data Compression Techniques

Xu Shubing

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Computer Engineering

Prince of Songkla University

2011

Thesis Title Hybrid Selective X-masking and X-canceling Multiple Input Signature Register for Test Data Compression Techniques

Author Mr. Xu Shubing

Major Program Computer Engineering

Major Advisor:

……
(Asst. Prof. Dr. Taweesak Reungpeerakul)

Examining Committee:

……..……..……..……..…….... .Chairperson
(Asst. Prof. Dr. Wannarat Suntiamorntut)

...……...
(Asst. Prof. Dr. Taweesak Reungpeerakul)

...……...
(Asst. Prof. Dr. Suntorn witosurapot)

...……
(Asst. Prof. Dr. Chayanoot Sangwichien)

……..
(Dr. Duenpen Kochakornjarupong)

The Graduate School, Prince of Songkla University, has approved this thesis as partial fulfillment of the requirements for the Master of Engineering Degree in Computer Engineering.

……

(Prof. Dr. Amornrat Phongdara)

Dean of Graduate School

Thesis Title	Hybrid Selective X-masking and X-canceling Multiple Input Signature Register for Test Data Compression Techniques
Author	Mr. Xu Shubing
Major Program	Computer Engineering
Academic Year	2011

ABSTRACT

For test response compaction in Circuit-Under-Test (CUT) with scan-based Design-For-Test (DFT)，the presence of Unknown-Values (X's) in test output responses during test can cause fault coverage lost and degrade the performance of test compression.

In this thesis, the present flexible X-masking logic called selective X-masking to handle the X's. The basic concept is to combine our X-masking logic with either X-canceling Multiple-Input-Signature-Register (MISR) or X-tolerant compactor. The selective X-masking is used to handle the majority of X's in scan chains, while the remained small number of X's can be tolerated by either X-canceling MISR or X-tolerant compactor. The experimental results based on ISCAS89 benchmark circuits have indicated that the presented selective X-masking logic can improve the compression ratio significantly and improve obviously the observability of scan cells.

Key words: X-masking, Test Response Compaction, MISR

ACKNOWLEDGEMENT

The thesis, together with the 3 years study here is coming to an end. At this moment, I would like to extend my sincere thanks to all those who have concerned about or helped the writing.

My deepest gratitude goes first and foremost to Asst. Prof. Dr. Taweesak Reungpeerakul, he gives instructive advice and useful suggestions to me on my thesis. His understanding, patience, encouragements have been the most important factors which increase my confidence extremely.

Second, I would like to express my heartfelt gratitude to all lecturers in department of Computer Engineering, who helped me a lot on my study. Additionally, I would like to thank the graduate School, the staffs of Faculty of Engineering and the staffs of Department of Computer Engineering for giving the convenience for me.

Third, I would like to thank my friends for their supports and assistances.

Finally, my thanks would go to my beloved family for their loving considerations and great confidence in me all through these years.

Xu Shubing

CONTENTS

Page

CONTENTS (CONTINUED)

Page

CONTENTS (CONTINUED)

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

LIST OF FIGURES (CONTINUED)

Figure Page

LIST OF FIGURES (CONTINUED)

Figure Page

LIST OF ABBREVIATIONS AND SYMBOLS

DFT	Design-for-Test
CUT	Circuit-Under-Test
ATE	Automatic-Test -Equipment
MISR	Multiple-Input -Signature -Register
SOC	System-On-Chip
RTL	Register-Transfer-Level
CAD	Computer-Aided -Design
BIST	Built-In-Self-Test
HCMR	Hierarchical-Configurable-Mask-Register
LFSR	Linear-Feedback-Shift-Register
VLSI	Very-Large-Scale-Integration
IC	Integrated-Circuit
PCB	Printed-Circuit-Board
VHDL	Very-High–Speed-Integrated-Circuit-Hardware-Description-Language
MUX	Multiplexer
TM	Test-Mode
SE	Scan-Enable
DI	Data-Input
SI	Scan-Input
DCK	Data-Clock
SCK	Shift-Clock
SRL	Shift-Register-Latch
LSSD	Level-Sensitive-Scan-Design
TPG	Test-Pattern-Generator
ORA	Output-Response-Analyzer
X’s	Unknown-Values
ATPG	Automatic-Test-Pattern-Generation
CR	Configuration-Register
MR	Mask-Register

LIST OF ABBREVIATIONS AND SYMBOLS (CONTINUED)

OP	Observation-Point
CK	Clock
CP	Control-Point

CHAPTER 1

INTRODUCTION

1.1 Motivation

The scan based test is one of the best approaches [1-3] in DFT which is used to increase the circuit's controllability and observability. The objective is to improve the testability of a design and to reach the target fault coverage goal. The scan test allows the test data coming from CUT to be stored on the Automatic-Test-Equipment (ATE) in a compressed form. The test data stored in ATE for stimulus compression and output response compaction determines the total test data overhead, and the test application time depends on the length of scan chains [4]. In order to reduce test cost in industry design, the initial long scan chains are also cut into large number of shorter scan chains. As the increasing huge test data volume in industry design, researches have recently being focused on input stimulus compression and output response compaction in order to reduce the number of test channels on the ATE, tester memory and test time.

The response compaction is implemented and received data from the outputs of the scan chains. The main purpose of the response compaction is to reduce the amount of test response transferred back to the ATE. A large number of test response compaction schemes have been proposed. Basically they include the space compaction [5] and the time compaction [6]. It is possible to combine time and space compaction, such as finite memory compactor [7] [8] which give advantage of time and space dimensions.

For the output response compaction, the presence of X's in the test responses has been the greatest barrier to effect the compaction. If there are no X's in the test responses, a time compactor, such as MISR, can compact an infinitely long output sequence into a fixed-length signature [9]. However, when X's appear in the test responses, it can cause an unpredictable signature, from which no faulty-circuit signature could be distinguished. When X's are introduced to the space compactors, the non-X test responses in the current clock cycle going

through the compactor are XORed by X's , then the non-X values are corrupted and the fault coverage might be lost. One of the major issues for test compaction is how to handle test responses containing X's. The sources of X's are caused by several conditions such as bus contention, uninitiated memory, un-modeled logic, floating tri-states bus, etc.

Lots of schemes have been proposed to handle X's in the output response. The widely used techniques to handle X's are X-blocking [10], X-masking [11-14] and X-tolerant [15-17]. The X-blocking scheme needs extra logic in the CUT [15], which can cause fault coverage lost and additional area overhead. Several X-masking techniques were proposed in previous work [11-13]. In one of X-masking techniques, the conventional Linear-Feedback-Shift-Register (LFSR) X-masking scheme [13] guarantees to mask all X's and keeps specified bits (d's) as well, where each value contained by d's is used to detect one or more faults. However, a large amount of mask data was generated for masking every scan slice. Another X-masking scheme is called reiterative X-masking [14], the volume of mask data decreases greatly due to reusing the mask bit. It is necessary to use interval counter for controlling.

Instead of masking X's, X-tolerant schemes have been introduced in [15] [17]. In [15], XOR gates are used to minimize the impact of non-X value being masked by X's. Since it can guarantee to check erroneous compactor output in the presence of limited number of X's, the errors can be detected by the tester when the errors are propagated to the compactor outputs with X's appearing in the current cycle. In the schemes of [17], there are N scan chains and the compacted test responses outputs are M, where N>M. In order to detect faults, the test responses from scan chains containing d's and X's are propagated to the different compactor outputs. However, the compaction ratio is extremely degraded as the increasing X's and the corrupted outputs by X's may decrease the fault coverage.

In this research, an efficient approach is developed based on the X-masking technique .It can reduce a large amount of mask bits without losing fault coverage in the presence of X's in test responses. The following main contents will be described from chapter 2 through chapter 5: the research background of compaction techniques and impact of X's in the test responses; the literature review on handling X's by focusing on several types of X-masking and

X-tolerant techniques; the proposed method; conclusion and discussion, respectively.

1.2 Objective

1) To reduce masking data overhead for test responses.

2) To improve the observability for scan cells in the presence of X's.

3) To improve the efficiency of X-masking when the distribution of X's has the tendency to be clustered in the output responses.

1.3 Scope of Work

1) Study the scan design in DFT method for Very-Large-Scale-Integration (VLSI) and the test architecture for test compression.

2) Study test response compaction focusing on X-masking and X-tolerant techniques.

3) Implement and analyze X-masking scheme.

1.4 Work Plan

1) To investigate and research on the basic idea of X-masking and X-tolerant schemes.

2) To find out an efficient X-masking technique for reducing the mask bits

3) To present the proposal.

4) To implement the reiterative X-masking scheme and compute the total overhead of mask bits based on the ISCAS89 benchmark circuits.

5) To compare the performance between the proposed scheme and previous appproaches.

6) To combine the selective X-masking for measuring the observability of scan cells.

7) To analyze implementation results and make a conclusion.

8) To submit the proposed papers in the international conferences and write the

final report.

1.5 Outline

This document is organized in 5 chapters as follow:

Chapter 1: Introduction. The motivation, objective and the scope of thesis are presented in this chapter. Then, the work plan for investigating the X-masking schemes is given as follow.

Chapter 2: Research Background. It includes the DFT architecture, scan test compression techniques and the problem statement in the presence of X's.

Chapter 3: Related schemes for handling unknown values. In this chapter, the wildly used schemes are described to handle X's in the output responses including X-masking and X-tolerant schemes. It discusses the issues about achieving high fault coverage and high compression ratio in the presence of X's.

Chapter 4: The proposed Selective X-masking and Results. A flexible X-masking logic is introduced in this section. The basic operation and principle based on the distribution of X's are presented.

Chapter 5: Conclusion and discussion.

CHAPTER 2

RESEARCH BACKGROUND

2.1 Introductory Testing Concepts for VLSI Circuits

Testing techniques for VLSI circuits are now facing many excited and complex challenges. As the continuously shrinking technologies in the large systems embedded in a single System-On-Chip (SOC), the right behavior of the whole systems is very important. The electronic testing consists of Integrated-Circuit (IC) testing, Printed-Circuit-Board (PCB) testing, and system testing at the various manufacturing stages during the whole system operation. The main function of testing is not only to find the fault-free systems but also to improve production yield at the various stages of manufacturing by analyzing the cause of defects when faults are encountered. In some systems, the periodic testing is implemented to ensure fault-free system operation and to initiate repair procedures once the faults are detected. Hence the VLSI testing techniques is very important for all the designers, product engineers, test engineers, manufacturers, and end-users [18].

Testing typically includes the parts of applying a set of test stimuli to the inputs of CUT and analyzing the output responses, as showed in Figure 2.1.Circuits that produce the correct output responses for all input stimuli are considered to be fault-free. The circuits that produce an incorrect response at any point during simulation are assumed to be faulty. Testing is performed in different stages in the lifecycle of VLSI device, such as during the VLSI development process, the electronic system manufacturing process, and, in some cases, system-level operation.

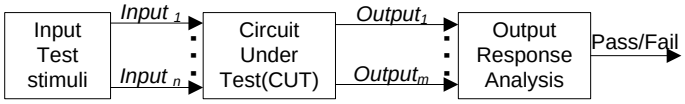


Figure 2.1 Basic testing approach [9]

For the VLSI development process, it can be seen that some form of testing is involved at each stage of the process in Figure 2.2. The VLSI device that fulfills the customer or project requirement is determined and formulated as a design specification. Design verification is a predictive analysis to ensure that the synthesized design can perform the function requirement when it is manufactured. Designers are responsible for synthesizing a circuit that satisfies the design specification and for verifying the design. Once a design error is found, the design is necessary to be modified and design verification must be repeated. As a result, design verification can be considered as a form of testing.

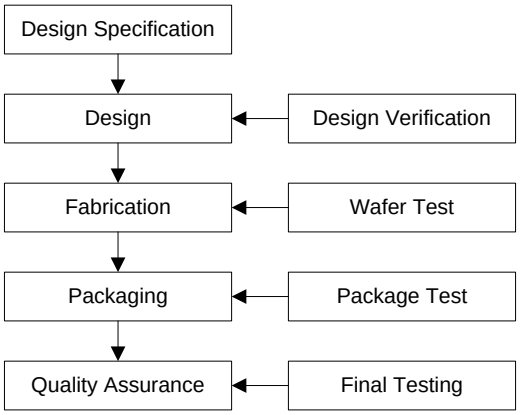


Figure 2.2 VLSI development process [9]

After that, then the VLSI design comes to fabrication. It is necessary to develop a test procedure based on the design specification and fault models at the same time. Since it is impossible for 100% of any particular kind of IC to be defect-free due to unavoidable statistical flaws in the materials, the ICs fabricated test on the wafer is the first test during the manufacturing process in order to determine which devices are defective. After passing the wafer-level test the chips are extracted and packaged. The packaged devices need to be retested because those devices may have been damaged during the packaging process or put into defective packages. The last test is the final testing before the chips go to market, including measurement of such parameters as input/output timing specifications, voltage, and current. In addition, stress or burn-in testing is often performed where chips are subjected to high temperatures and supply voltage. In a word, the design verification is very important for the VLSI development process and

even for the whole VLSI testing.

For the design specification, it can be divided into several levels, as shown in Figure 2.3. The design process is always transformed from a higher level description to lower level description. The initial level is a behavioral level which is developed in Very-High –Speed-Integrated-Circuit-Hardware-Description-Language (VHDL) or Verilog or as a C program and simulated to determine if it is functionally equivalent to the specification. After that, the design comes to Register-Transfer-Level (RTL) level description, which is verified with the functionality of the behavioral level and performed with more structural information including the data paths and control circuits. The logic level is synthesized from the RTL description and is designed to guarantee the correct functionality. For the physical level description, the physical placement is obtained and the transistors in the VLSI device are interconnected to fabrication.

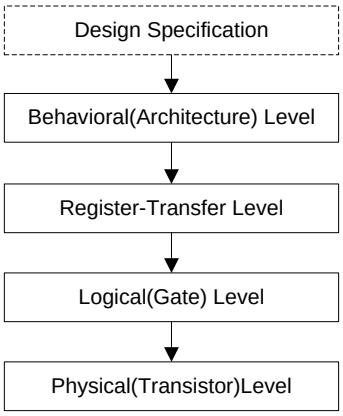


Figure 2.3 Design hierarchy [9]

Many tools have been developed for design verification process such as Computer-Aided-Design (CAD), synthesis, hardware emulation, and formal verification methods. However, design verification takes time, and insufficient verification fails to detect design errors. Thus, the process of the design verification is extremely important. Moreover, the test stimuli are often applied to design verification of the RTL, logical, and physical levels for testing the VLSI device.

Normally, the following two undesirable situations may occur after ICs are tested:

1. A faulty device appears to be a good part passing the test.
2. A good device fails the test and appears as faulty.

These two situations happen often due to a poorly designed test or the lack of DFT. For the first case, even if all products pass acceptance test, some faulty devices can be found in the manufactured electronic system. The next section will give the basic background for DFT.

2.2 Introduction for DFT

A substantial amount of time and effort is required when test engineers usually have to construct test vectors after the design is completed. However, the effort can be avoided if testing is considered early in the design flow to make the design more testable. Hence, integration of design and test, referred to as DFT, was first proposed in the 1970s. In order to test circuits, we need to control and observe logic values of internal lines. However, it would be very difficult to control and observe some nodes in sequential circuits. Testability measures of controllability and observability were first defined in the 1970s, which is considered to find some parts of a digital circuit that will be most difficult to test in test pattern generation for fault detection. Many DFT techniques have been proposed since that time as generally falling into three categories: (1) Ad hoc DFT techniques, (2) Scan design (3) Built-In-Self-Test (BIST).

2.2.1 Ad hoc DFT techniques

Ad hoc methods were the first DFT techniques proposed in the 1970s [1]. It targets only those portions of the circuit that would be difficult to test. Ad hoc DFT techniques typically involve applying good design practices or replacing a bad design practice with a good one. Table 2.1 lists some typical ad hoc techniques. One of the most widely used techniques is test point insertion. The conception is to insert test point directly to access internal nodes to improve the controllability or observability.

Table 2.1 Typical *Ad hoc* DFT Techniques

A1	Insert test points
A2	Avoid asynchronous set/reset for storage elements
A3	Avoid combinational feedback loops
A4	Avoid redundant logic
A5	Avoid asynchronous logic
A6	Partition a large circuit into small blocks

The observation point insertion for a logic circuit with three low-observability nodes is shown in Figure 2.4. Observation-Point (OP) shows the structure of an observation point that consists of a Multiplexer (MUX) and a D flip-flop. It includes the Scan –input (SI) and Scan-Output (SO). A low-observability node is connected to the port '0' of the MUX for an observation point, and all observation points are serially connected into an observation shift register using the port '1' of the MUX. An Scan-Enable (SE) signal is applied to select MUX port. The multiplexer uses SE input to select between the Data-Input (DI) and the Scan-Input (SI) .When *SE* is set to 0 and the Clock (CK) is applied, the logic values of the low-observability nodes are captured into the D flip-flops. When *SE* is set to 1, the D flip-flops within OP1, OP2, and OP3 operate as a shift register, allowing us to observe the captured logic values through OP_output during sequential clock cycles. As a result, the observability of the circuit nodes is greatly improved.

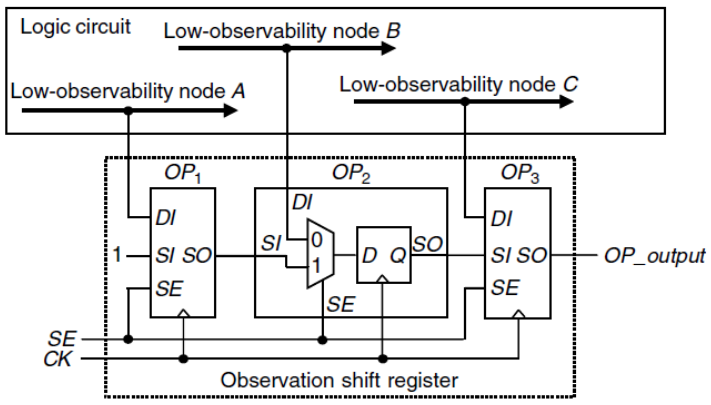


Figure 2.4 Observation point insertion [9]

Figure 2.5 shows an example of control point insertion for a logic circuit with three low-controllability nodes. The Control-Point (CP) is composed of a MUX and a D flip-flop. It includes the basic SI and SO. The original connection at a low-controllability node is cut, where a MUX is inserted between the source and destination ends. During normal operation, the Test-Mode (TM) is set to 0 and the CK is applied, the value drives from the source end to the destination end through the port'0' of the MUX. During test model, TM is set to 1 so that the value from the D flip-flop drives the destination end through the port'1' of the MUX. The D flip-flops in OP1, OP2, and OP3 are designed to form a shift register so the required values can be shifted into the flip-flops using CP_input. Hence, the controllability of the circuit nodes is greatly improved. However, the control point insertion can result in additional delay to the logic path. So the control points can't be inserted on a critical path.

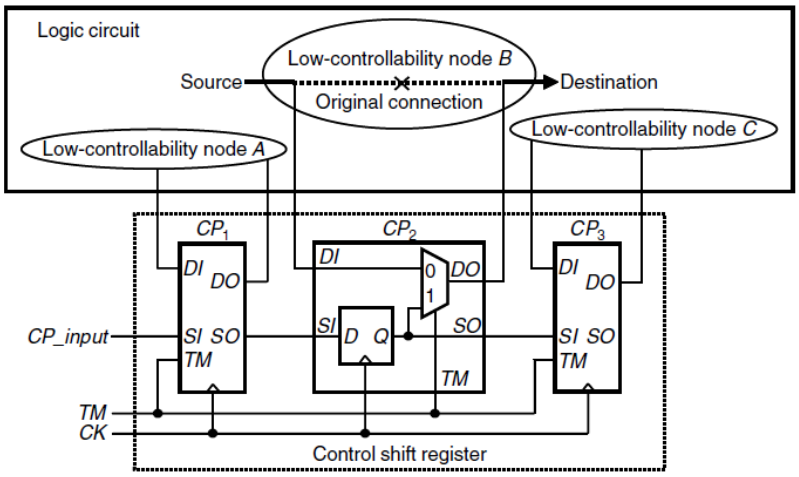


Figure 2.5 Control point insertion [9]

2.2.2 Scan Design

There are basically two input sources in a scan cell. The first input, data input, is driven by the combinational logic of a circuit. The second input, scan input, is driven by the output of another scan cell in order to form one or more shift registers called scan chains. The scan chain is performed by connecting the scan input of the first scan cell to a primary input and the output of the last scan cell to a primary output. In order to allow a scan cell to operate in two different modes: normal/capture mode and shift mode, a selection mechanism must be provided.

In normal/capture mode, data input is selected to update the output. In shift mode, scan input is selected to update the output.

The Muxed-D scan design is one of the most widely used schemes in logic design. The basic operation is o pass a logic value from its input to its output when a clock is applied. The muxed-D scan cell design is shown in Figure 2.6(a). It consists of a D flip-flop and a multiplexer. The multiplexer uses SE input to select between the Data-Input (DI) and the Scan-Input (SI). During normal/capture mode, *SE* is set to 0, and the value in the input DI is captured into the internal D flip-flop when a rising clock edge is applied. In shift mode, SE is set to 1. The SI is now used to shift in new data to the D flip-flop while the content of the D flip-flop is being shifted out. Sample operation waveforms are shown in Figure 2.6(b).

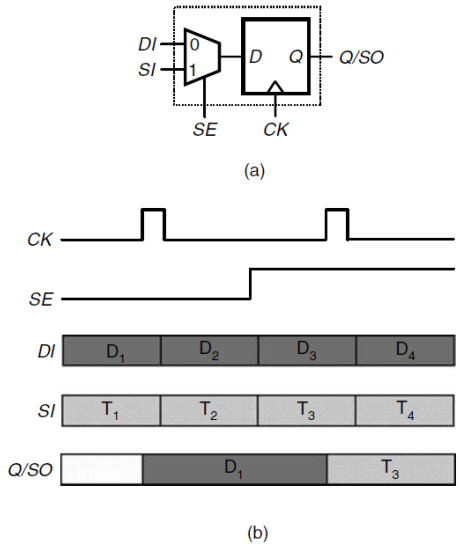


Figure 2.6 Edge-triggered muxed-D scan cell design and operation [9]

A clock-scan cell design was another important approach proposed in [2]. Comparing with a muxed-D scan cell, a clocked-scan cell also has a data input DI and a scan input SI; however, the input selection in the clocked-scan cell is applied with two independent clocks, Data-Clock (DCK) and Shift-Clock (SCK), as shown in Figure 2.7(a). In normal/capture mode, the present value at the data input DI is captured into the clocked-scan cell by data clock DCK. During shift mode, the shift clock SCK is used to shift in new data from the scan input SI into the clocked-scan cell, while the current content of the clocked-scan cell is being shifted out. Sample

operation waveforms are shown in Figure 2.7(b). The main advantage is that it can result in no performance degradation on the data input. However, it requires additional shift clock routing.

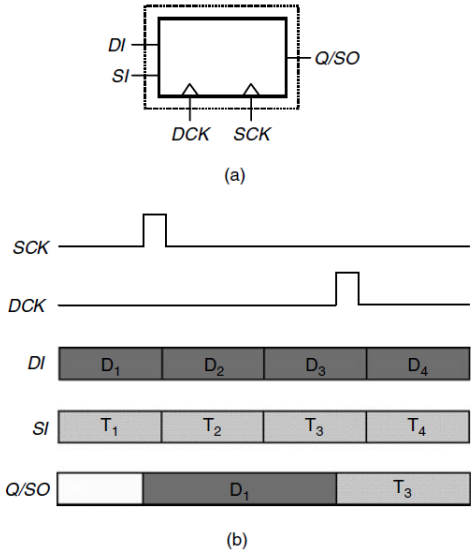


Figure 2.7 Clocked-scan cell design and operation [9]

Different from muxed-D scan cells or clocked-scan cells using for edge triggered, flip-flop-based designs, the Level-Sensitive-Scan-Design (LSSD) scan cell is used for level-sensitive, latch-based designs [19] [20] [21]. Figure 2.8(a) shows a polarity-hold Shift-Register-Latch (SRL) design described in [19] that can be used as an LSSD scan cell. This scan cell consists of two latches, a master two-port D latch L_1 and a slave D latch L_2. Clocks C, A, and B are used to select between the data input D and the scan input I to drive $+L_1$ and $+L_2$, which can be used to drive the combinational logic of the design. In capture mode, master latch L1 uses the system clock C to latch system data from the data input D and to output this data onto $+L_1$, and clock B is used after clock A to latch the system data from latch L_1 and to output this data onto $+L_2$. In shift mode, clocks A and B are used to latch scan data from the scan input I and to output this data onto$+L_1$, and then latch the scan data from latch L_1 and to output this data onto $+L_2$. Sample operation waveforms are shown in Figure 2.8(b).

The main advantage of LSSD is to insert scan into a latch-based design. But the technique requires routing for the additional clocks, which increases routing complexity.

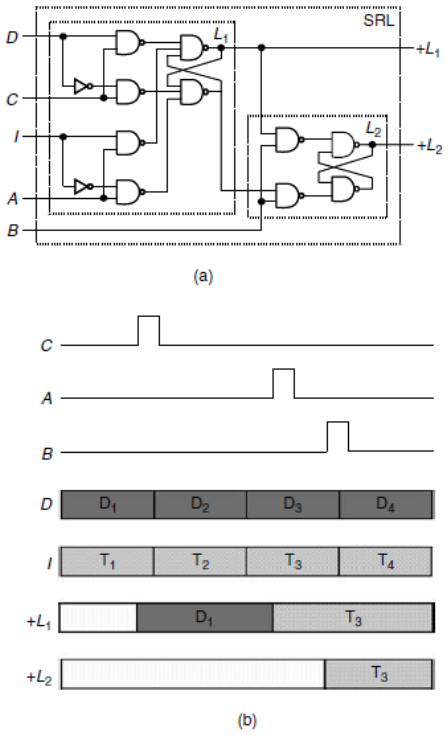


Figure 2.8 Polarity-hold SRL design and operation [9]

2.2.3 Built-In-Self-Test (BIST)

BIST was proposed around 1980 [3] [22], which integrate a Test-Pattern-Generator (TPG) and an Output-Response-Analyzer (ORA) in the VLSI device to perform testing internal to the IC, as illustrated in Figure 2.9. The test pattern can be generated automatically by TPG for application to the inputs of the CUT. The ORA is then applied to compact the output responses of the CUT into a signature. The logic BIST controller can generate specific BIST timing control signal for coordinating the BIST operation among the TPG, CUT, and ORA. During the BIST operation, the compacted final signature needs to compare with an embedded golden and the logic BIST controller then provides a pass/fail indication. The compaction for output responses requires that all storage elements in the TPG, CUT, and ORA must be initialized to known state, which means that no X could be propagates from the CUT to the ORA.

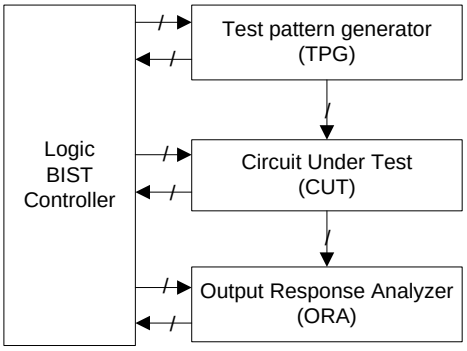


Figure 2.9 Typical logic BIST system [9]

Scan DFT techniques have been wildly used in the industry. Some of major elements for scan test cost are: (1) test data volume, which translates to tester memory requirement; (2) total number of tester channels;(3) test time, which translates to the maximum number of flip flops in a scan chain. As the IC chip complexity increases, test data volume also increases rapidly. Since test data volume is a major factor that determines test cost, several test compression techniques to reduce both volume of test patterns and output responses have been developed, which is described in next.

2.3 Test Compression Techniques

For the test compression, it involves the amount of compressing data including both stimulus and response that is stored on ATE for testing with a deterministic test set. The benefit of test compression is that the amount of test data can achieve a 10× or even 100× reduction on the ATE. The ATE memory requirements are greatly reduced and even more importantly it reduces test time because less data has to be transferred across the bandwidth between the ATE and the chip. Moreover, the test compression techniques are easy to implement in industry design because they are compatible with the conventional design rules and test generation flows used for scan testing.

Figure 2.10 shows the test data bandwidth between the tester and the chip. As ATE has limited speed, memory, and I/O channels, the chip cannot be tested any faster than the amount of time required to transfer the test data, and the time is equal to:

$$\frac{\text{Amount of test data on tester}}{(\text{Number of tester channels})(\text{Tester clock rate})}$$

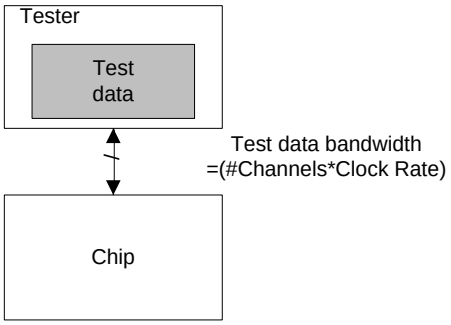


Figure 2.10 Block diagram illustrating test data bandwidth [9]

Test compression technique is to compress the test data stored on ATE including both stimulus and responses, and the structure is illustrated in Figure 2.11. It can largely reduce the total amount of tester memory. Moreover, the test time can be obviously reduced due to less test data transferred across the low bandwidth link between the tester and the chip. It shows that the additional on-chip hardware is integrated before the scan chains to decompress the test stimulus coming from the tester and after the scan chains to compact the response going to the tester.

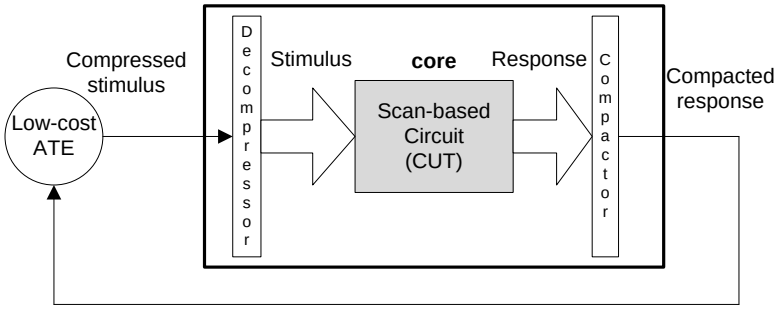


Figure 2.11 Architecture for test compression [9]

2.4 Test Stimulus Compression

Test data is inherently highly compressible. In fact, typically only 1 to 5% of the bits have specified values, and majority of unspecified bits that are not assigned values during Automatic-Test-Pattern-Generation (ATPG). Consequently, test stimulus compression can be used to significantly reduce the amount of test stimulus data that must be stored on the tester. Normally, ATPG procedures perform random fill, in which all the unspecified bits in the test cubes are filled randomly with 1's and 0's to create fully specified test vectors. During test stimulus compression procedure, the specified (care) bits should be lossless in order to preserve the fault coverage of the original test cubes. After decompression, the resulting test patterns shifted into the scan chains should match the original test cubes in all the specified bits. Many compressing schemes have been proposed, which can be classified into the three categories: (1) Code-based techniques, (2) Linear-decompression-based schemes (3) Broadcast-scan-based schemes.

2.4.1 Code-Based Schemes

In order to encode the test cubes, the data compression codes partition the original data into symbols which can be replaced with a codeword to form the compressed data. The decompression is performed by using a decoder that simply converts each codeword into the corresponding symbol. Depending on whether the size of symbols and codewords is fixed or variable, data compression codes can be classified into four categories.

Take a dictionary code (fixed-to-fixed) for example, which had been proposed in [23]. There are n-bits blocks to form the symbols in the original test cubes, and then the symbols are encoded with codewords that each has b bits, where b<n. One can view each symbol through dictionary because each codeword has been indexed into the dictionary that points to the corresponding symbol. Since there are 2n possible for symbols and 2b possible for codewords, no all possible symbols can be in the dictionary. If S_{data} is the set of symbols that occur in the original data, the number of distinct symbols that occur in the original data $|S_{data}|$ is much less than 2^n. The compression ratio is equal to:

$$2^{n-\lceil \log_2 |s_{data}| \rceil} : 1$$

The scheme is shown in Figure 2.12. There are n (n>=1) scan chains, and the test cubes are partitioned into n-bit symbols such that each scan slice corresponds to a symbol. The size of each codeword is b bits, where b = $\lceil \log_2 |s_{data}| \rceil$ .This figure shows that the b channels from the tester can be used to load n scan chains. Normally, b channels can just load b scan chains from the tester. Thus, the length of each scan chain becomes shorter and less clock cycles are required. The dictionary code gives a good example for how test compression reduces not only tester storage but also test time.

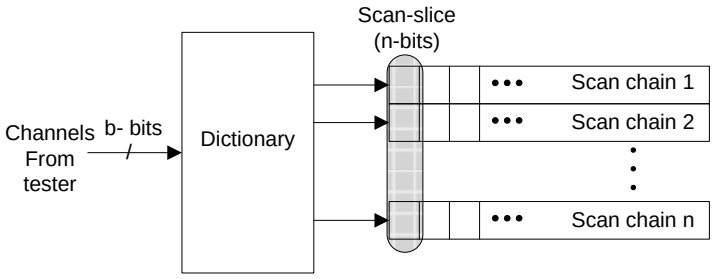


Figure 2.12 Test compression using a complete dictionary [9]

2.4.2 Linear-Decompression-Based Schemes

Another test stimulus compression scheme is to expand the original data from the tester to scan chains by using linear decompressor, which consists of only XOR gates and flip-flops. The linear space can be spanned by a Boolean matrix. In other words, the linear decompressor can expand an m-bit compressed stimulus from the tester into an n-bit test vector, the set of test vectors that can be generated by the linear decompressor is spanned by A, and a Boolean matrix is $A_{n\times m}$. Any test vector Z can be compressed by a particular linear decompressor if and only if there exists a solution to a system of linear equations, **A**X = Z, where **A** is the characteristic matrix of the linear decompressor and X is a set of free variables stored on the tester. There is an example of a sequential linear decompressor shown in Figure 2.13, where each free variable coming from the tester is represented by a symbol. The initial state of the LFSR is represented by the free variables X_1−X_4, and the free variables X_5−X_{10} are shifted in by two channels as the scan chains are loaded. After symbolic simulation, the characteristic matrix for a linear decompressor can be obtained and the final values in the scan chains are represented by the

equations for Z_1−Z_{12}.

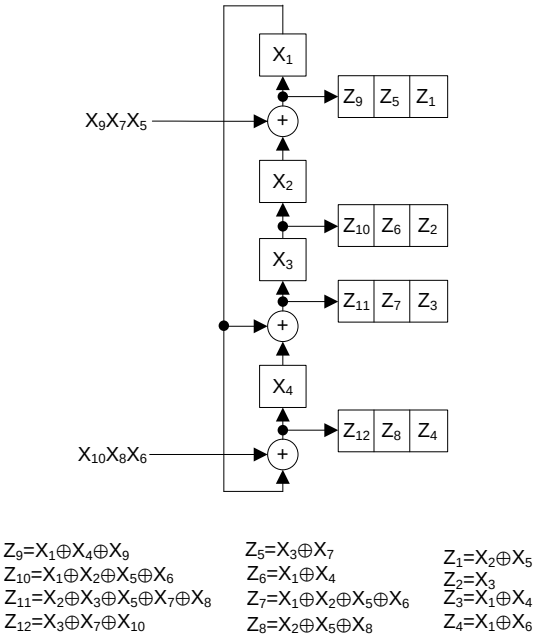


Figure 2.13 Example of symbolic simulation for linear decompressor [9]

The operation of symbolic simulation is shown as follow: Assume that the initial seed X_1−X_4 have been already loaded into LFSR. In the first cycle, the top flip-flop is filled by the XOR of X_2 and X_5, the second flip-flop is filled by X_3, the third flip-flop is filled by the XOR of X1 and X4, and the bottom flip-flop is filled by the XOR of X_1 and X_6. Finally, it gets $Z_1= X_2⊕X_5$, $Z_2= X_3$, $Z_3= X_1⊕X_4$, and$Z_4= X_1⊕X_6$.

In the second cycle, the top flip-flop is filled by the XOR of X_3 and X_7, where X_3 is the contents of the second flip-flop; the second flip-flop is filled by the values of the third flip-flop ($X_1⊕X_4$); the third flip-flop is filled by the XOR of the values of the first flip-flop ($X_2⊕X_5$) and the fourth flip-flop ($X_1⊕X_6$); and the bottom flip-flop is filled by the XOR of the values of the first flip-flop ($X_2⊕X_5$) and X_8. Thus, it can get $Z_5= X_3 ⊕X_7$, $Z_6= X_1⊕X_4$, $Z_7= X_1⊕X_2⊕X_5⊕X_6$, and $Z_8= X_2⊕X_5⊕X_8$. In the third cycle, the top flip-flop is filled by the XOR of the values of the second flip-flop ($X_1⊕X_4$) and X_9; the second flip-flop is filled by the values of the third flip-flop ($X_1⊕X_2⊕X_5⊕X_6$); the third flip flop is filled by the XOR of the values of the first flip-flop ($X_3⊕X_7$) and the fourth flip-flop ($X_2⊕X_5⊕X_8$); and the bottom flip-flop is filled by

the XOR of the values of the first flip-flop ($X_3 \oplus X_7$) and X_{10}. Thus, it gets $Z_9 = X_4 \oplus X_9$, $Z_{10} = X_1 \oplus X_6$, $Z_{11} = X_2 \oplus X_5 \oplus X_8$, and $Z_{12} = X_3 \oplus X_7 \oplus X_{10}$. At the moment, the scan chains are fully loaded with a test cube and the whole simulation is complete. The corresponding system of linear equations for this linear decompressor is shown in Figure 2.14.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \\ X_8 \\ X_9 \\ X_{10} \end{bmatrix} = \begin{bmatrix} Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \\ Z_5 \\ Z_6 \\ Z_7 \\ Z_8 \\ Z_9 \\ Z_{10} \\ Z_{11} \\ Z_{12} \end{bmatrix}$$

Figure 2.14 System of linear equations for the decompressor [9]

For the linear decompressor, the linear equations can be used to encode the test cube and can be solved with the Gauss–Jordan elimination [24]. The test cube that does not find a solution is called unencodable test cube, In this case, it is not possible to encode the test cube with this particular linear decompressor. In order to handle unencodable test cubes, a simple way is just to bypass the decompressor when applying those test cubes. Another approach is to restart the ATPG to find a different test cube that is encodable. A third approach is to use more free variables when decompressing the test cubes, which can make it easier to solve the linear equation. Generally, it is very unlikely to be able to encode a test cube as it has more specified bits than the number of free variables. In other words, when the number of free variables is sufficiently larger than the number of specified bits, the probability of not being able to encode the test cube becomes negligibly small. For example, if the number of free variables is 20 more than the number of specified bits, then the probability of not finding a solution is less than 10−6. The encoding efficiency, for linear decompressors can be defined as follows:

$$(\text{Specified Bits in Test Set})/(\text{Bits Stored on Tester})$$

The optimality of encoding efficiency is 1. However, it is not possible to achieve higher than an encoding efficiency of 1 because there are normally more free variables than specified bits.

Since a sequential linear decompressor allows free variables from earlier clock cycles to be used when encoding a scan slice in the current clock cycle, it can provide much more flexibility than combinational decompressors and solve the problem of the worst-case most highly specified scan slices limiting the overall compression. In other words, the more flip-flops that are used in the sequential linear decompressor, the greater flexibility can be achieved. A typical design of a sequential linear decompressor were described in [25], [26], and [27], as illustrated in Figure 2.15. There are b channels from the tester that load free variables into LFSR (one of linear finite-state machines), and the following is the combinational linear XOR network that expands the outputs of the LFSR to fill scan chains. In decompressing process for each test cube, the state of the LFSR is first reset and then some initial free variables are loaded to LFSR within a few clock cycles. After that, additional free variables are loaded into the LFSR in each clock cycle. The total number of free variables that are use to fill each test cube is equal to b (q+m), where b is the number of channels from the tester, q is the number of clock cycles used to initially load the LFSR, and m is the length of the longest scan chain.

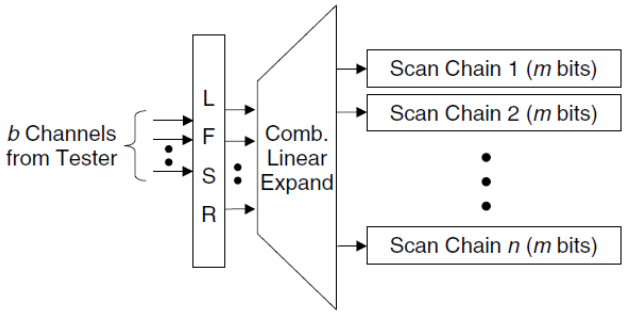


Figure 2.15 Typical sequential linear decompressor [9]

2.4.3 Broadcast-Scan-Based Schemes

The third kind of test stimulus compression schemes that broadcast the same value to multiple scan chains was first proposed in [28] and [29]. This method has been widely used in many test compression architectures. The advantage is the same pattern can be extended to multiple circuits as illustrated in Figure 2.16. The problem is how to guide the ATPG tool to generate the patterns to be shared. Generally, the inputs here include the primary inputs as

well as the pseudo-primary inputs.

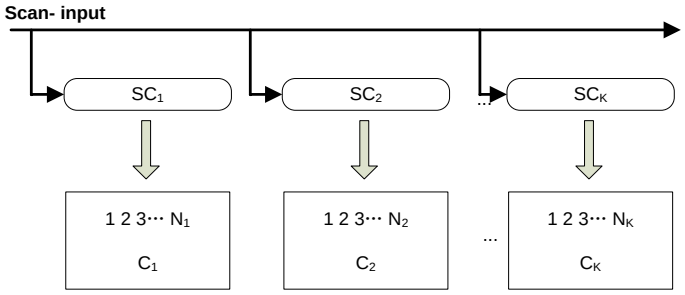


Figure 2.16 Broadcasting to scan chains driving independent circuits [9]

Since one test vector can detect a fault in a stand-alone circuit then it will still be possible to apply this vector to detect the fault in the broadcast structure, the main benefit of broadcast scan for independent circuits is that all faults that are detectable in all original circuits will also be detectable with the broadcast structure. Thus, the broadcast scan method will not affect the fault coverage if all circuits are independent. Moreover, broadcast scan can also be applied to a single circuit if all subcircuits driven by the scan chains are independent.

After the test stimulus data has been applied to CUT, the response data requires to be shifted out for compaction. The compaction technique is described next.

2.5 Test Response Compaction

Due to a large number of scan cells in complex designs, the test data volume would be large and the test time would be long. A common way for reducing test data volume and test time is to break original scan chains into a larger number of shorter scan chains, and then use a smaller number of ATE channels to specify inputs stimulus and observe output responses. Figure 2.17 shows the architecture for input stimulus compression and output response compaction in a scan-based design. It contains total k test channels from ATE for input stimulus, where (k<n). After compaction, the total bits are m, which is greatly less than the total number of scan chains (n). The output response compaction can generate a “signature” for output responses, the signature then can be observed through a limited number of channels to compare with the good-circuit signature. In order to support a large number of scan chains with a limited number of

ATE channels, researchers have recently worked on input stimulus compression and output response compaction.

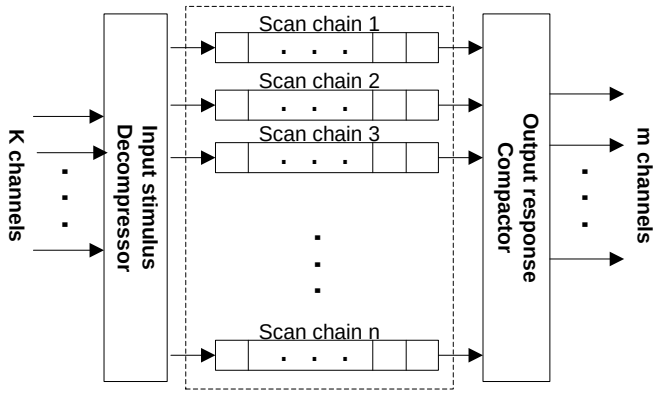


Figure 2.17 Compression scheme for scan-based designs

Test response compaction is performed at the outputs of the scan chains. The test data after compaction must be transferred back to the tester. Unlike lossless for test data in stimulus compression, the test responses in test compaction can be lossy. The compaction techniques generally are classified into three categories: space compaction, time compaction, and mixed time and space compaction.

2.5.1 Space Compaction

The space compaction can be expressed as a linear function of the input vector X and the output vector Y

$$Y = \varphi(X)$$

where X is an m-bit input vector and Y is an p-bit output vector, p<m, asillustrated in Figure 2.18.

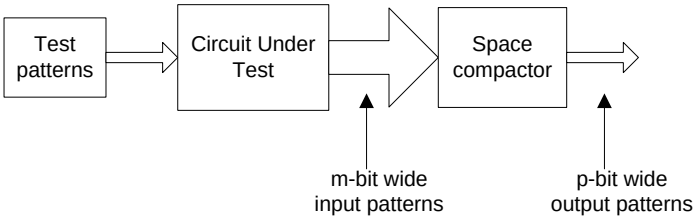


Figure 2.18 Space compaction schemes [9]

Figure2.19 is a simple example to show the space compactor for compacting 4

(m=4) scan chains of the circuit under test to 1(p=1) output, where the combinational circuit consists of the network of XOR gates.

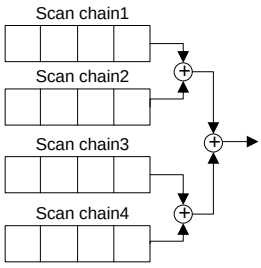


Figure2.19 Basic space compactor structure

2.5.2 Time Compaction

Different from space compactor using combinational logic, the time compactor uses sequential logic. A time compactor compacts n input patterns to q output patterns (where q < n), as illustrated in Figure 2.20. Once time compactor is applied, no X's from circuit under test is allowed to reach the compactor.

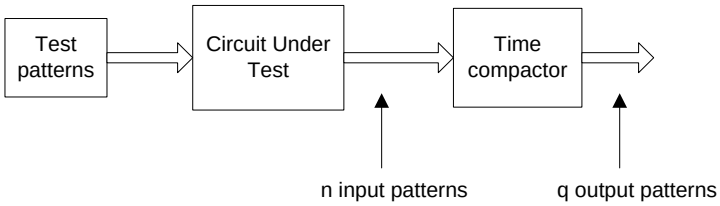


Figure 2.20 Time compaction schemes [9]

The most widely used time compactor is the MISR. The Figure 2.21 demonstrates the n-stage MISR. The structure of the n-stage MISR can be expressed by a specified characteristic polynomial of degree n, f(x), the symbol h_i can be set either 1 or 0 based on the existence or absence of the feedback path, where:

$$F(x) = 1+h_1x+h_2x^2 + \cdots+h_{n-1}x^{n-1}+x^n$$

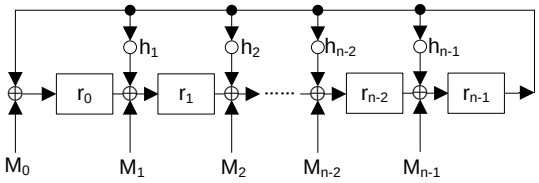
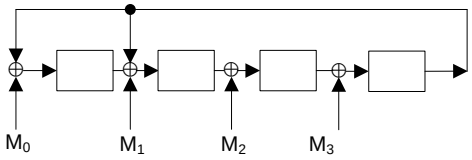


Figure 2.21 An *n*-stage MISR [9]

Take an example for the four-stage MISR applying $f(x) = 1+x+x^4$, as illustrated in Figure 2.22(a). Set M_0= {10010}, M_1= {01010}, M_2= {11000}, and M_3= {10011}. After calculating by the formula $M(x) = M_0(x) + xM_1(x) + x^2M_2(x) + x^3M_3(x)$, it can get $M(x) = 1+x^3+x^4+x^6+x^7$ or M= {10011011}, as shown in Figure 2.22(b). Finally, the signature R is represented by the rightmost four bits of the M sequence. Hence, R= {1011}.



(a)

M_0	1 0 0 1 0
M_1	0 1 0 1 0
M_2	1 1 0 0 0
M_3	1 0 0 1 1
M	1 0 0 1 1 0 1 1

(b)

Figure 2.22 An equivalent *M* sequence for four-stage MISR [9]

In this example, there are totally four input patterns (n=4) such as M_0, M_1, M_2, and M_3. After compaction, there is only one output pattern (q=1) R.

2.5.3 Finite Memory Compaction

In this section, finite memory compaction is introduced, which combines the advantages of time compaction and space compaction. Many finite memory compactors have been proposed such as OPMISR [30], convolutional compactor [31], and q-compactor [32] [33] [34].

Figure 2.23 shows a simple example about the basic structure of a q-compactor, including six inputs coming from scan chains and one output, where each flip-flop is

connected to the corresponding output of XOR network. It shows that every error in a scan cell can reach storage elements and then outputs in several possible ways. Since the q-compactor does not have feedback path, any error can be shifted out for comparing with the expected data after at most five cycles.

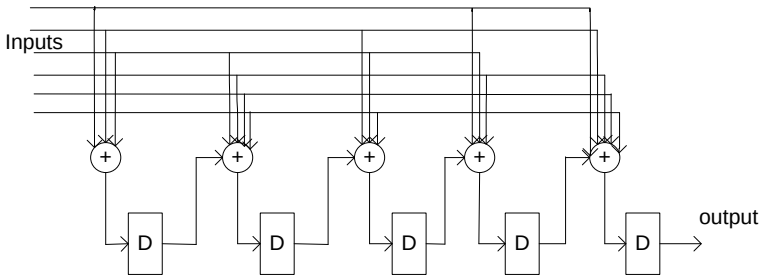


Figure 2.23 An example *q*-compactor with single output [9]

The list three types of compactor is very efficient for output compaction. However, once X's occur in output responses, the fault coverage may lost due to error masking, which is described in next section.

2.6 Error Masking

Once these X's are introduced into the output response data, the performance of the test compression schemes can be significantly degraded. For the class of output compactors called time compactors, MISRs for example, once an unknown is introduced, the signature quickly becomes corrupted and must be reset, that means no error can be observed. For example, the MISR and two scan chain in Figure 2.24. The scan chain 1 contains one error in first cycle and scan chain 2 contains an X in second cycle respectively, after two cycles the ε and μ are XORed together in MISR bit, then the error is corrupted.

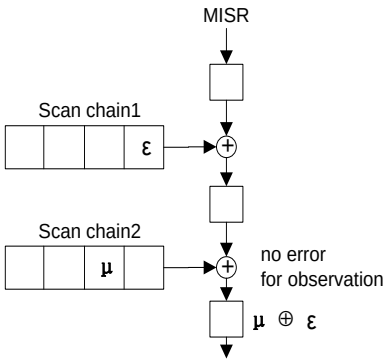


Figure 2.24 Error masking in MISR

When the scan-out responses come to output compactors, the scan-out responses are compacted by XOR gates. However, an error response may not be observed due to unknown-induced error masking. An error means a response different from the good circuit response. An X means the value in response is not determined in simulation. Take an example for space compactor in Figure 2.25, an error and an X are represented by 'ε' and 'μ', respectively, and 's' means don't care bit. The unknown and error can be propagate to the output of XOR operation, for example the result of XOR operation between 'μ' in scan chain 1 and 's' in scan chain 2 is 'μ'. The error cannot be observed after XOR operation with an X, and the second stage of XOR operation shows the result of XOR operation between 'ε' and 'μ'. This is called unknown-induced error masking which happens when there is an X along with some errors at the inputs of the compactor. Under such a situation, the good-circuit response at the compactor output is X, Thus, no error can be observed.

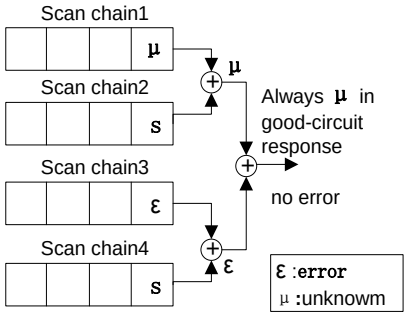


Figure 2.25 Error masking in space compactor

Since the structure of the finite memory compactors is adding some memory elements in the space compactor, the compaction schemes obtained by combining time and space

dimensions, so the error masking in finite memory compactors happens in both situation of time compactor and space compactor**.**

2.7 Summary

In this chapter, the overview of VLSI testing technology development is provided and the main challenge of VLSI testing is discussed. In order to illustrate the problem statement clearly, the whole picture is draw in this area including the basic DFT concepts, scan design and test compression techniques. As shown in this chapter, the presence of X's in test responses can greatly degrade the output compaction due to the error masking. In order to handle this problem, several efficient schemes are proposed in the next chapter.

CHAPTER 3

RELATED SCHEMES FOR HANDLING UNKNOWN VALUES

3.1 Introduction of Schemes for Handling X's

Scan design has become main stream for complex digital circuits in test industry. For the test compression, the main goal is to achieve high test quality with lower test overhead. However, the presence of X's in test streams can largely degrade the efficiency of compression during test mode. To reduce the impact of X's, a large number of schemes for handling X's have been developed in recent years. The basic methods are X-blocking, X-masking and X-tolerant.

During response capture, the generated X's can be blocked from reaching the scan cell. One way is to identify X sources and add additional DFT logic to fix the X sources by adding additional test points [35]. Another way is to block the X's from reaching the scan cells by careful test pattern generation. The approach in [36] is to identify the X-candidates and to generate control vectors for reducing the number of X's in the scan response. X-blocking can ensure that no X's will be observed. However, it does not provide a means for observing faults, which can cause fault coverage lost. In addition, the approach in [36] is not compatible with the test data compression technique.

As the X-blocking technique may cause fault coverage lost and it does add area overhead and may impact delay due to the inserted logic, the X's can also be masked off right before the response compactor [37-40]. The X-masking technique can be easily implemented as it just adds the X-masking logic in front of compactor. It does cause additional hardware overhead. However, it does not cause any circuit delay during the test.

Meanwhile, the X's can be tolerated during the compaction of the test response. The X-tolerant technique is used to design a compactor which can guarantee the errors are detected even in the presence of X's in the same clock cycle. The main advantage is it does not

require any additional hardware for X-masking logic and does not degrade the system performance during normal operation. For the space compactors, it utilizes XOR gates to compact test response coming from N scan chains $\{0, 1, X\}^N$ into a compacted observable test response with Q outputs $\{0,1,X\}^Q$ for each scan cycle, where N > Q. The following section lists several related schemes for handling X's in recent years.

3.2 X-masking Techniques

As mentioned earlier, the X-masking techniques are applied to mask X's before the X's coming to response compactor. Figure3.1 shows a general structure of X-masking circuit with three scan outs. It shows that the mask controller can apply a logic value '1' for any scan output to mask off X's in scan output responses. Notice that the mask date can be generated for each scan cycle and it can be stored in compressed format.

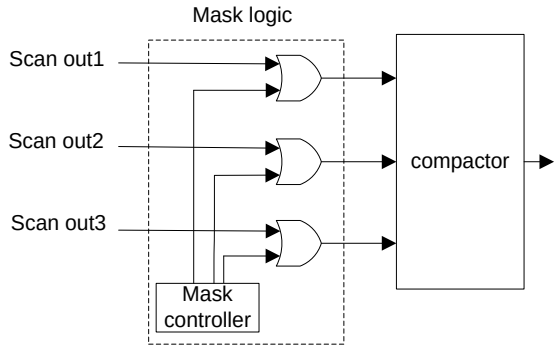


Figure 3.1 A simple X-masking circuit

Many schemes for X-masking have been proposed. In this section, some important X-masking schemes are list, including conventional LFSR X-masking, reiterative X-masking, and X-masking by use of a Hierarchical-Configurable-Mask-Register (HCMR).

3.2.1 Conventional LFSR X-masking

The conventional LFSR X-masking is wildly used scheme which is described in [13]. It creates a mask for every scan slice to ensure that all X's in each scan slice get masked,

and all d's remain unmasked, where d records a value that is required to be observed for the detection of one or more faults. The basic architecture of this technique is shown in Figure 3.2. We can see the LFSR is used to drive scan chains and create the mask data from the phase shifter as well. The compressed mask data stored on the ATE is expanded through LFSR for each scan slice. Since the ATE has to store enough mask data to create a mask for every slice, amounts of mask data are required.

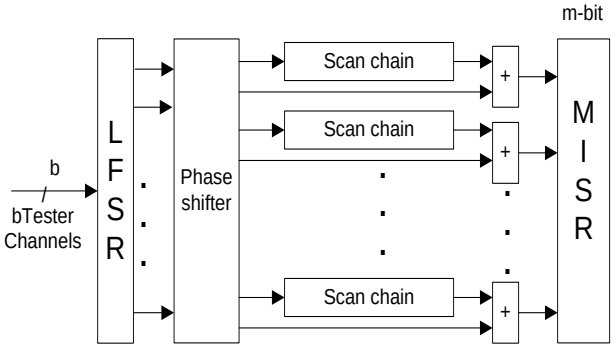


Figure 3.2 Architecture of conventional LFSR X-masking [13]

Figure 3.3a shows the basic operation for conventional LFSR X-masking. For an interval of each scan slice, the mask bit is given for each scan cell. The mask bit can be set to '1' (masked) if *X* is present in a scan cell ; the mask bit can be set to '0' (unmasked) if *d* is present; if there is neither of *X* and *d* in a scan cell, the mask bit can be set to '?'(don't care). The results are shown in Figure 3.3b where none of *X's* is left and all *d's* remain after masking.

Scan Slice	Scan chains 0	1	2	3	4	5	6	7	8	9
5	s	d	s	**x**	s	s	**x**	s	s	s
4	s	**x**	s	s	s	d	s	s	s	s
3	s	s	**x**	s	d	d	s	s	s	s
2	s	d	s	s	s	s	**x**	s	s	s
1	s	d	s	s	s	**x**	s	s	s	s
mask										
5	?	0	?	1	?	?	1	?	?	?
4	?	1	?	?	?	0	?	?	?	?
3	?	?	1	?	0	0	?	?	?	?
2	?	0	?	?	?	?	1	?	?	?
1	?	0	?	?	?	1	?	?	?	?

(a)

	0	1	2	3	4	5	6	7	8	9
5	s	d	s	s	s	s	s	s	s	s
4	s	s	s	s	s	d	s	s	s	s
3	s	s	s	s	d	d	s	s	s	s
2	s	d	s	s	s	s	s	s	s	s
1	s	d	s	s	s	s	s	s	s	s

(b)

Figure 3.3 Conventional LFSR X-Masking Example

As the conventional LFSR X-masking scheme creates X-mask for each scan cell, it ensures that all X's can be masked off, but the number of mask bit is huge. In order to reduce the mask bit overhead, a further research is given in next section.

3.2.2 Reiterative X-masking

The reiterative X-masking technique can create a mask for more than one scan slice, as described in [14]. This technique lists two efficient methods, including variable reiterative LFSR X-Masking and hybrid X-masking approach based on different number of X's in test response.

3.2.2.1 Variable Reiterative X-masking

The variable reiterative LFSR X-Masking scheme applies a mask for a number of scan slices. It also ensure that all X's in each scan slice get masked and all d's get unmasked, but less number of mask data are used by finding a mask that is usable for a varying number of adjacent scan slices. The structure of this approach is shown in Figure 3.4. The decompressor is used to load scan chains and create mask data every *i* clock cycles.

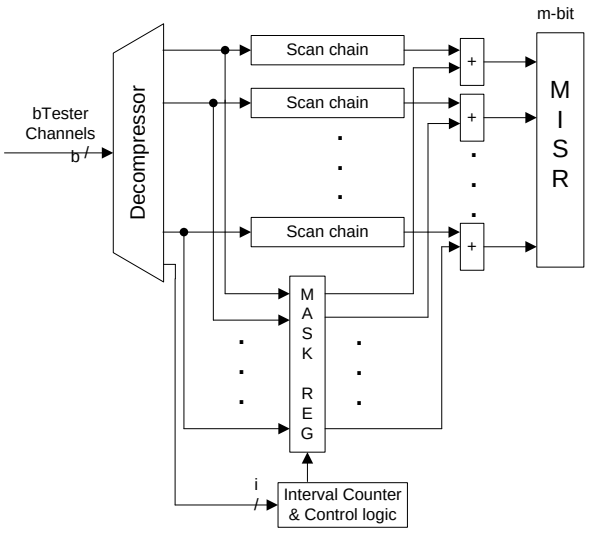


Figure 3.4 Structure of variable reiterative LFSR X-Masking [14]

The way for finding a solution of the masks for a given pattern is list as follow:

1) First find the location of all the *d* and *X* bits.
2) Apply a mask for the first scan slice at the beginning of the current interval. For example, the scan slice 1 in Figure 3.5a gets an X in position 6, so the mask bit for this position is '1'. As positions 1 and 2 in scan slice 1 are *d*'s, so the mask bits for those positions are '0's.
3) Add the interval by one to load the current mask. For scan slice 2, bit position 5 gets a *d*, so the mask bit for this position is set to '0'. Since position 4 gets an X, the mask bit is '1'.
4) Rerun step 3 until at least one mask bit position gets a conflict. If the conflict happens, the current scan slice needs to be set the beginning of a new interval. Then skip to step 2. For example, when load scan slice 5 to the current interval, an X occurs in bit positions 2 and 5, which can cause a conflict. Finally, M1 creates the mask for scan slices 1-4, and M2 is for 5-7. Figure 3.5b shows the results after masking.

Scan Slice	Scan chains 0	1	2	3	4	5	6	7	8	9
7	s	d	s	s	s	**x**	s	s	s	s
6	s	s	s	d	d	s	d	s	**x**	s
5	s	d	**x**	s	s	**x**	s	s	s	s
4	s	s	s	d	s	d	s	s	s	s
3	s	s	d	s	s	s	**x**	s	s	s
2	s	s	s	s	**x**	d	s	s	s	s
1	s	d	d	s	s	s	**x**	s	s	s
M2	?	0	1	0	0	1	0	?	1	?
M1	?	0	0	0	1	0	1	?	?	?

(a)

	0	1	2	3	4	5	6	7	8	9
7	s	d	s	s	s	s	s	s	s	s
6	s	s	s	d	d	s	d	s	s	s
5	s	d	s	s	s	s	s	s	s	s
4	s	s	s	d	s	d	s	s	s	s
3	s	s	d	s	s	s	s	s	s	s
2	s	s	s	s	s	d	s	s	s	s
1	s	d	d	s	s	s	s	s	s	s

(b)

Figure 3.5 Variable Reiterative LFSR X-Masking Example [14]

As it can reuse masks for multiple adjacent scan slices, the amount of mask data stored on the ATE can be reduced significantly. Thus, the output data compression can be greatly improved. However, this approach is efficient just for small numbers of X's. When the number of X's gets large, a hybrid X-masking is proposed in next section.

3.2.2.2 Hybrid X-masking Approach

Hybrid X-masking approach uses fixed-interval reiterative X-masking combined with conventional LFSR X-masking. The primary advantage of the reiterative *X*-masking logic is to mask all of the easy to mask *X*'s by reusing masks for many scan slices, and the conventional LFSR *X*-masking can handles the rest. The structure of this approach is shown in Figure3.6.

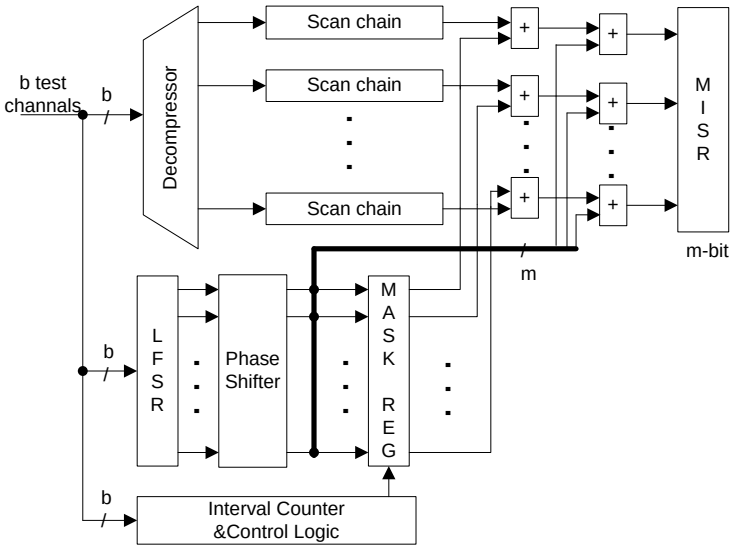


Figure 3.6 Hybrid X-Masking architecture [14]

In this Figure, the decompressor is used to drive scan chains and the LFSR with phase shifter is applied to generate the mask bits. For *m* scan chains, the hardware architecture consists of an *m*-bit masking register, an *i*-bit interval counter, 2*m* OR gates. This LFSR is not only used to generate the mask for each slice for the conventional LFSR X-masking part of this approach, but also generate the mask bits to load the mask register for the reiterative

X-masking part. Reiterative LFSR X-masking logic is designed to reuse a mask data when adjacent scan slices got *X's* mask *X's* for a number of scan slices and keep all *d's*. In order to keep the overhead of storing the mask data small, a reseedable LFSR is used to encode just the *X* and *d* bits. The interval counter is loaded one time at the beginning of the test from the *b* tester channels. The interval counter counts down the number of shift cycles until the interval counter becomes zero. Once that happens, a new mask is loaded into the mask register and the interval counter is returned to the initial value.

The operation for fixed-interval reiterative X-masking is shown in Figure 3.7a, For an interval of five scan slices, mask bits can be set to '1' (masked) in scan chains 2, 3, and 6 if X's are present without d's, and mask bits can be set to '0' (unmasked) in scan chains 1, 4, and 5 since d's are present. If there are neither of X's and d's in scan chains 0, 7, 8, and 9, the mask bits is set to '?'(don't care). The results are shown in Figure 3.7b where in two X's are left in scan chains 1 and 5, which can be handled by conventional X-masking logic. Note that the size of the interval counter determines the overall effectiveness of this hybrid approach. If the size of interval counter gets smaller, the remaining of X's usually presents less, but more mask bits are required, otherwise the mask bits will be smaller. Thus, the minimum size of interval counter has been determined and can be re-loaded in different test patterns.

Scan slice	Scan Chains 0	1	2	3	4	5	6	7	8	9
5	s	d	s	**x**	s	s	**x**	s	s	s
4	s	**x**	s	s	s	d	s	s	s	s
3	s	s	**x**	s	d	d	s	s	s	s
2	s	d	s	s	s	s	**x**	s	s	s
1	s	d	s	s	s	**x**	s	s	s	s
Mask	?	0	1	1	0	0	1	?	?	?

(a)

	0	1	2	3	4	5	6	7	8	9
5	s	d	s	s	s	s	s	s	s	s
4	s	**x**	s	s	s	d	s	s	s	s
3	s	s	s	s	d	d	s	s	s	s
2	s	d	s	s	s	s	s	s	s	s
1	s	d	s	s	s	**x**	s	s	s	s

(b)

Figure 3.7 Reiterative X-Masking Example [14]

Comparing conventional LFSR X-masking, this approach can provide

significant reduction of mask bits overhead as masks are reused for multiple scan slices. Moreover, this approach handles the X's into two cases: small number of X's and large number of X's. Using Variable Reiterative LFSR X-masking can significantly increase the amount of compression for smaller numbers of *X*'s. When the number of X's gets large, then a hybrid approach is applied. In order to mask X's more specifically, another X-masking scheme called hierarchical configurable mask register is proposed to apply optimal mask bit based on the different distribution of X's in the next.

3.2.3 Hierarchically Configurable Register

In this section, a hierarchical method is proposed in [41] for X-masking by use of a HCMR. The main idea of this approach is to select a subset of *l* of the *k* scan chains first. Then an optimized X-mask is applied only for selected *l* scan chains in each test pattern. A single control signal is used to determine whether the X-mask will be applied or not for every scan cycle. After the X-mask operation, the remaining X-values can be tolerated by an X-tolerant compactor. The basic implement of this approach is described as follow.

3.2.3.1 Implement of Hierarchical Configurable Mask Register

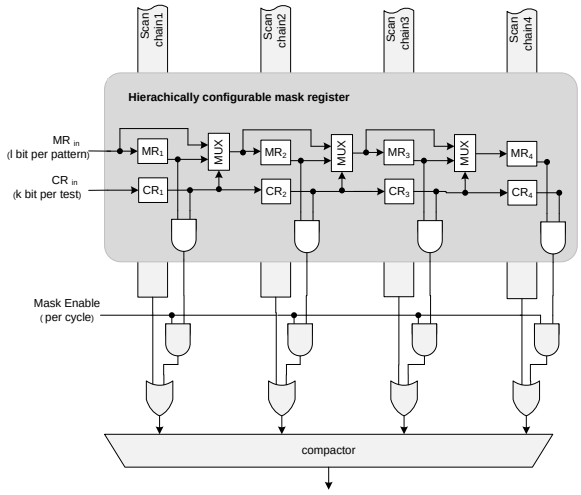


Figure 3.8 Structure of hierarchical configurable mask register [41]

The hardware structure of the proposed *HCMR* is described in Figure 3.8. There are four (*k=4*) scan chains, and the HCMR have two separate shift register Configuration-Register (CR) and Mask-Register (MR) in it. The *CR* consists of flip-flops $CR_1,...,CR_k$ and *MR* consists of flip-flops $MR_1,...,MR_k$. A multiplexer MUX_i is connected to the every output of MR_i. The *Mask Enable* signal is used to allow the activation of X-mask. Once *Mask Enable* signal is active, the selected scan chains are allowed to be masked based on the values stored in *MR*.

Figure 3.9 shows a simple integration of the *HCMR*. There is a single module of *HCMR* for one scan chain, as shown in Figure 3.9a. The configuration register *CR* selects a subset of *l* scan chains. Since the *i-th* bit of configuration register CR_i is set to logic 'true', the i-th scan chain needs to be masked. If the logic is set to 'false', it means the scan chain is no need to be masked. Note that the configuration register *CR* is only loaded one time at the beginning of test.

For the each pattern, the mask can be shifted into mask register *MR* for *l* selected scan chains. The case whether or not the *i-th* scan chain is masked is determined by the value of output m_i which is provided by an AND gate from MR_i and CR_i. Therefore, the value of MR_i can be propagated to output m_i if and only if the value of CR_i is 'true'. This approach is efficient as the load process of mask register MR loads only *l* bits of mask bit. The multiplexer MUX_i is controlled by the value of CR_i. Once the value in CR_i is logic 'true', the value stored in MR_i is propagated to multiplexer output. Otherwise the value at the input of MR_i is propagated to multiplexer output. The both cases are shown in Figure 3.9b and Figure 3.9c, respectively. Thus, it just needs to load *l* positions to *MR* because the *CR* only defines *l* bit for *HCMR*.

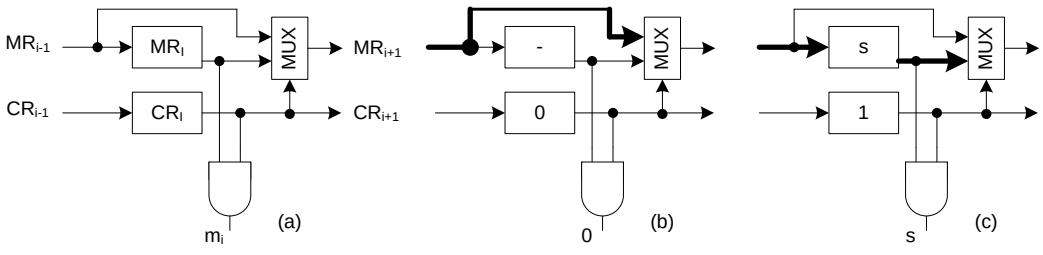


Figure 3.9 Mask register element for a single scan chain [41]

Figure 3.10 shows a simple example to demonstrate HCMR with k=4. As the value in CR_1 and CR_4 are logic 'true', the mask register just generates two mask bits s_1 and s_4 for the first and last flip-flop. Thus, the mask bit is only loaded to the first and fourth scan chain as the output m_2 and m_3 are logic 'false'. The *HCMR* is required two tester channels for controlling. The first channel is used to transfer the information for configuration register *CR* and mask register *MR*. the second is use to drive the *Mask Enable* signal.

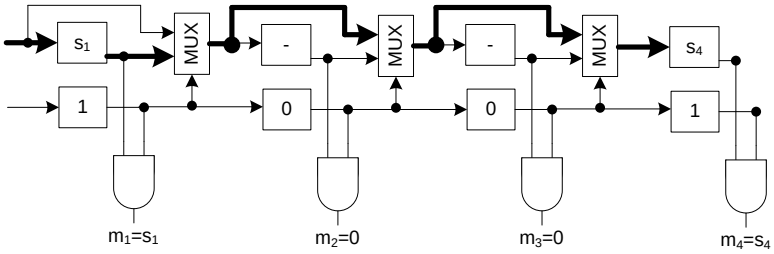


Figure 3.10 4-stage mask register [41]

After understand the implement of hierarchical configurable mask register, the next step is to apply an optimal value for register *CR* and *MR* to achevie a high observability of scan cells.

3.2.3.2 Optimal X-mask Determination

In this section , a nearly optimal X-masking for a test pattern is proposed. For the first step, the configuration of register *CR is* determined by *l*, where *l* also represents the number of selected scan chains. If the masks are not sufficiently efficient, the number of l can be increased to 2*l*,3*l*,.... Note that the configuration select *l* scan chains with most X's during the whole test. Only the bits CR_i corresponding to these *l* scan chains are set to logic 'true'. The set values are unchangeable during whole test.

For the second step, it defines the optimal X-mask M(m, p) for each pattern p, and X-mask M(m,p) selects m ($0<m<l$) scan chains with most X's. However, the optimal X-mask M(m,p) needs to be iteratively determined. For convenience, the value m is modified by the value *s* ($0 \leq s \leq l$)for every iteration . Define C_M : the number of observable scan cells using the X-mask M(m,p) and C_{notM}: the number of observable scan cells without X-mask. So the additional number

of observable scan cell is $\Delta c = C_M - C_{notM}$.

For the basic operation of the iterative process, it set the *s* a fix value first if additional number of observable scan cell Δc increases for each iteration. Once Δc decreases, the value of *s* is reduced by half for each iteration. The iteration process requires to stop as long as the same X-mask is generated. Finally, several X-mask M(m, p) are produced for one test pattern. The best X-mask which gets the largest amount of additional observable scan cells can be selected.

The iterative process can be illustrated in Figure3.11 and Table 3.1. In Table 3.1, the initial value of *s* is set to10 for the first iteration. As the value of Δc increases , the value of *s* is unchanged within first two iterations. For the third iteration, the value of Δc reduces comparing with the second iteration. So the value of *s* reduces half. For the iteration process of five and six ,the value of Δc increases again. The iteration process will stop after the sixth iteration because the value of m=25 has been generated twice. Comparing six X-mask for whole iteration process, the maximum additional observable scan cell can be achieved in second iteration, where Δc is 90. Thus ,the final X-mask is to select m=20 masked scan chains for this pattern.

Table 3.1 Example of iterative process

Iteration	1	2	3	4	5	6	7
Δc	50	90	70	60	80	85	60
s	10	10	10	5	2	1	1
m	10	20	30	25	23	24	25

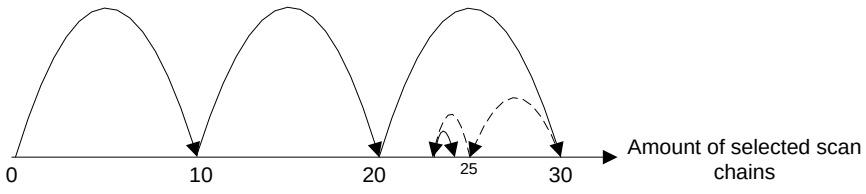


Figure 3.11 Development of the amount of masked scan chains during iterative process [41]

This proposed X-mask scheme a *HCMR* is very efficient as it utilizes the fact that most of X's are only located in a subset of scan chains. As mentioned before, the mask logic is

performed in two steps. For the first step, a subset of scan chains is selected to consider for X-mask. The second, an optimal X-mask is applied to the *HCMR* for each test pattern. The *HCMR* can be easily integrated in arbitrary designs.

In order to detect fault, one valid way is to mask off X's directly by specified X-masking logic in the list X-masking techniques. However, the fault can also be detected without using X-masking logic in the presence of X's, which is presented in X-tolerant scheme in next section.

3.3 X-tolerant Schemes

Since the time compactor such as MISR cannot tolerate even one X, it is necessary to develop an X-masking scheme to mask all the X's in test response. However, another type compaction scheme such as space compactor can allow the existence of X's, so it is more suitable to develop an X-tolerant scheme for space compactor in general. The main advantage of X-tolerant schemes is that it does not need to generate mask bits to mask X's and it causes negligible hardware overhead.

Instead of masking X's directly by specified mask logic in X-masking technique, the X-tolerant scheme is performed to minimize the impact of non-X value being masked by X's. It can guarantee to check erroneous compactor output in the presence of limited number of X's, or the errors can be detected by the tester when the errors are propagated to the compactor outputs with X's appearing in the current cycle. In this section, several important X-tolerant schemes are presented, such as response shaper, X-canceling MISR, X-compactor and i-compactor.

3.3.1 Response Reshape

In this section, a circuit module called response shaper is proposed in [4], which can reduce the masking effect due to X's for space compactor. In Figure 3.12, we can see the response shaper logic is added in front of space compactor, so that the scan-out responses need to be "reshaped" first by response shaper logic before compaction. After the proper control signals

based on the simulation results applied to configure the response shapers, the space compactor can detect faults more easily. Thus, the compaction scheme can improve obviously the fault coverage as well as the number of observable scan-out responses in the presence of unknown values(*X's*).

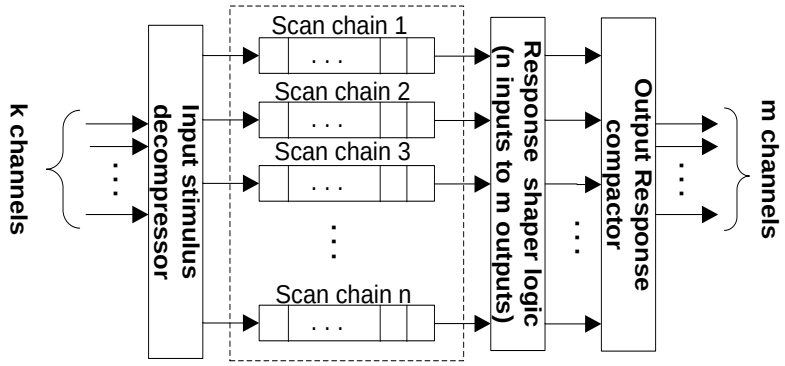


Figure 3.12 Input compression and output compaction scheme with response shaper [4]

The objective of this work is to improve the tolerance of *X's* by adding the response shaper logic to "reshape" the scan-output responses before compaction, which means ATE can detect as many modeled faults as possible in the presence of *X's*. Meanwhile, this scheme is applied to maximize the number of observable scan-out responses in the presence of *X's*. The basic operation response shaper is shown in next section.

3.3.1.1 Design of a Response Shaper

There is an example of a 4-scan-chain response shaper shown in Figure 3.13. Each input in_i and output out_i of this response shaper correspond to the scan chain *i*. In the response shaper logic, a 2-to-4 decoder is applied to select scan chain by assigning a proper value at signal *chain sel*. Since a scan chain *i* is not selected by the decoder, out_i will directly connect to the first scan cell in scan chain *i* (denoted as $C_{i,1}$). For a selected scan chain *i*, if *shi f t_sel* is equal to 0, *outi* will connect to a delay element which stored the scan-out response in the previous cycle. If *shi f t_ sel* = 1, *outi* will connect to a constant value (1 in this example). The operation (*shi f t_ sel* = 0) can be called the *shift operation* and the operation (*shi f t_ sel* = 1) is *replacement operation*.

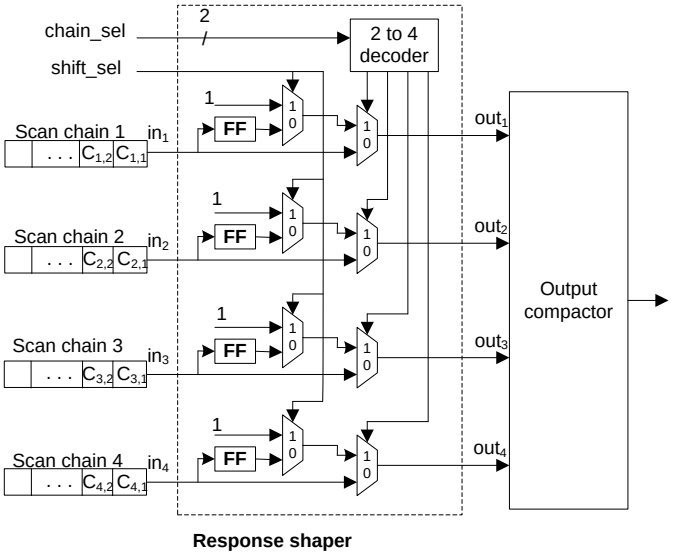


Figure 3.13 An example of a 4-scan-chain response shaper [4]

Figure 3.14 demonstrates the operation for rearranging the scan-output responses going to the space compactor after a shift operation and a replacement operation. $SO_{i,j}$ represents the scan-out response of scan cell $C_{i,j}$. For the scan chain i, there is neither a shift operation nor a replacement operation. Therefore , the scan-out responses coming to the space compactor for scan chain i are $SO_{i,1}$, $SO_{i,2}$, $SO_{i,3}$ and $SO_{i,4}$ in order in Figure 3.13 (a). After a shift operation applied on scan chain i ,the order is changed to $SO_{i,4}$, $SO_{i,1}$, $SO_{i,2}$, and $SO_{i,3}$,as shown in Figure 3.14 (b), where $SO_{i,4}$ is the same as $SO_{i,4}$ propagating to the space compactor in the previous clock cycle. In Figure 3.14 (c) shows all scan-out responses become a constant as a replacement operation is applied on scan chain i.

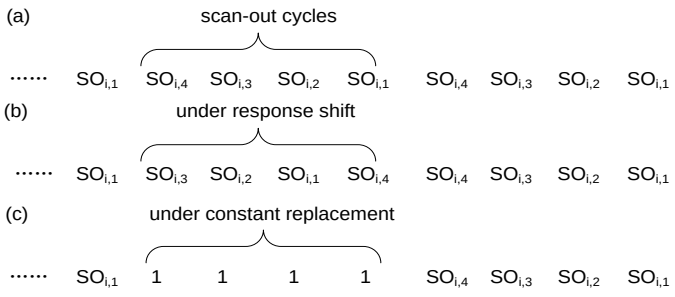


Figure 3.14 An example of shift operation and the replacement operation

By the way, a shift operation can perform not only a one-cycle delay. With more storage elements and its corresponding MUX, a shift operation can perform a multiple-cycle delay. However, such flexibility needs more control bits for *shi f t_ sel*. In order to further reduce the data overhead of the control signals applied to a response shaper, we do not change the signals of *chain_ sel* and *shi f t_ sel* during the entire scan-out cycles for one test pattern. For the following parts, the detail of fault detection and improvement for observable scan cells with a response shaper is described.

3.3.1.2 Fault Detection with a Response Shaper

Once values are applied to the control signals *chain_ sel* and *shi f t _sel*, the main issue of control signal generation is to maximize the fault coverage. A fault could be detected if one of its error responses is propagated to an output of the space compactor. Figure 3.15 shows an example of applying a response shaper to avoid the masking effect. In the scan-out responses, there are three errors and three *X's* produced in simulation. Without shift operation, after the XOR operation, none of these errors can be observed due to either even-error masking (the third scan-out cycle) or unknown-induced error masking (the second scan-out cycle).As a shift operation is applied on the first scan chain, the scan-out response will change to Figure 3.15 (b). In this situation, one error can be observed in the third cycle. Hence, the shift operation or replacement operation may help to detect some faults. But sometimes, some detectable faults may become undetectable after a shift operation or a replacement operation. So the reshaping algorithm for control signal generation is important.

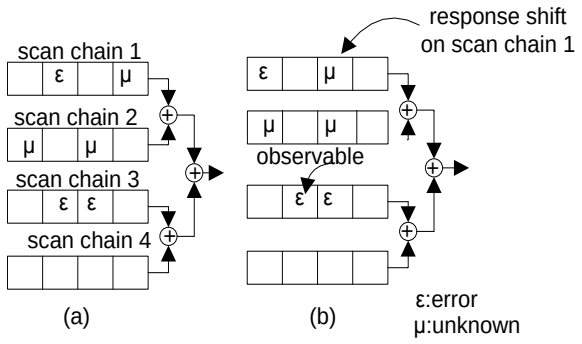


Figure 3.15 An example of fault detection with a response shaper [4]

3.3.1.3 Observable scan-out Response with a Response Shaper

Another application for control signal generation is to improve the percentage of observable scan-out responses. The definition of observable scan-out response is a scan-out response does not be masked by an unknown-induced error masking. For example, a scan-out response on a scan chain is observable if there is no *X's* on any other scan chain fed to compactor at the same cycle. The basic operation for increasing the number of observable scan-out responses is described in Figure 3.16. At first, the scan-out responses in $C_{1,3}$, $C_{2,3}$,$C_{3,3}$, and$C_{4,3}$ are observable due to no *X's* in current cycle, as shown in Figure 3.16(a). After a shift operation applied on the first scan chain in Figure 3.16(b), four more scan-out responses in $C_{1,1}$, $C_{2,1}$, $C_{3,1}$, and $C_{4,1}$ become observable as the *X's* in $C_{1,1}$ is shifted to $C_{1,2}$. When a replacement operation is applied on the first scan chain in Figure 3.16(c), six more scan-out responses in $C_{1,2}$, $C_{3,2}$, $C_{4,2}$, $C_{1,4}$, $C_{3,4}$, and $C_{4,4}$ can become observable because the two *X's* on the second scan chain are replaced by a constant 1. However, the scan-out response in $C_{2,3}$ is no longer observable because it has already been replaced by a constant 1.

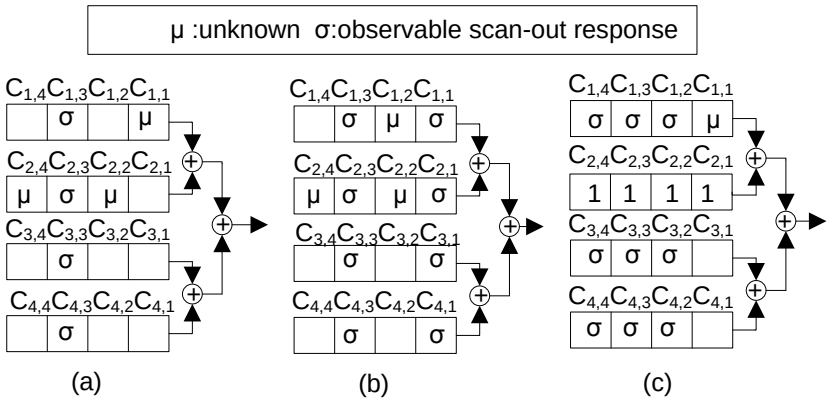


Figure 3.16 An example for observable scan-out response [4]

This scheme proposes a simple circuit module called the response shaper to enhance the *X's* tolerance for output response compaction. With the response shapers, the faults detection can be maximized for a space compactor. Meanwhile, the reshaping algorithm is important for control signal generation.

The method for enhancing the X's tolerance in this scheme is to reduce the

occurrence of error masking (including even error masking and unknown- error masking) for test responses. However, another method for enhancing the X's tolerance is to cancel out unknown value by itself, which is described in next section.

3.3.2 X-Canceling MISR

In this section, a *X*-tolerant MISR compaction methodology called *X*-Canceling MISR is introduced in [15]. Unlike conventional *X*-masking schemes, it does not require any masking logic at the input of the MISR. The basic idea is that each *X* in the output stream is represented by a unique symbol, which is used to determine the final state of the MISR. Each bit of MISR signature corresponds to a linear combination of symbols, and some linear combinations are guaranteed to be linearly dependent in terms of the symbols corresponding to the *X's* when there are more bits of MISR than symbols. Once some combinations of MISR bits which are linearly dependent the symbols corresponding to the *X's* are XORed together, the *X's* can cancel out each other and produce a deterministic value during the test. After comparing the value of the XORed combination of MISR bits and the fault-free value, an error can be detected if the values mismatch each other. The *q* such combinations can provide the error coverage of approximately $1\text{-}2^{-q}$. Therefore, if 7 such combinations are checked, the error coverage is given over 99%. Checking 16 such combinations can provide the error coverage 99.998%. To apply an appropriate set of MISR bit combinations for checking, it is possible to guarantee 100% coverage for some fault model. By the following part, a symbolic simulation is described.

3.3.2.1 Symbolic Simulation

After the output response that has been captured in the scan chains, each scan cell can be represented by a symbol. For example, in Figure 3.17, assume each symbol X_i has an *X* value and each symbol O_i has a non-*X* value. The final state of the MISR in terms of the symbols has been obtained since the output response is shifted in to the MISR. Finally, each bit of the MISR is equal to a linear combination of the scan cells. For example in Figure 3.17, the final value of the top bit of the MISR will be equal to $X_1 \oplus O_3 \oplus O_8 \oplus O_{13}$.

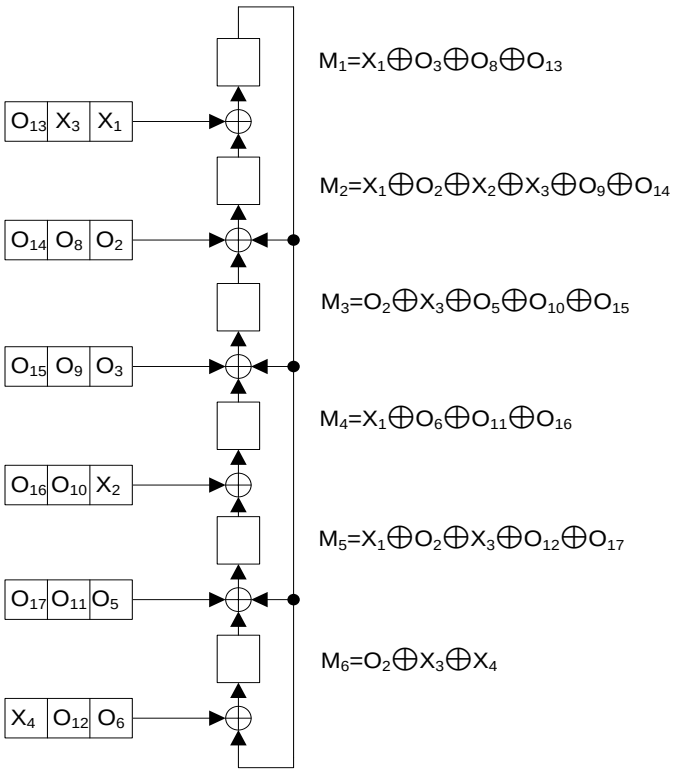


Figure 3.17 Example of Symbolic Simulation of MISR [15]

The Figure 3.17 shows the *X* dependence of the MISR bits, where each *X* take on value 0 or1. If there are *k X*'s, then there are totally 2^k different combinations for the *X*'s. Take an example in figure 2, there are 4 *X*'s in the output response $(X_1\text{-}X_4)$, and each could be either 0 or 1 in a fault-free circuit. So there are 16 possible signatures in all produced in the MISR for a fault-free circuit. Each possible combination of values for the *X's* can be thought of as producing a valid fault-free signature in the MISR. If an *m*-bit MISR contains *k X's*, which means 2^k valid signatures out of 2^m possible signatures can be produced in a fault-free circuit. Once an error occurs, it will change a fault-free signature to a faulty signature. So the probability for the faulty signature which belongs to the *2k* valid fault-free signatures is $2^k /2^m$. Therefore, the probability of aliasing is $2^k/2^m$. If *k* is 20 less than *m*, the probability of aliasing is 2^{-20} which less than one in a million. It means that an *m*-bit MISR can compact at most (*m-20*) *X*'s in output streams with negligible loss of error coverage.

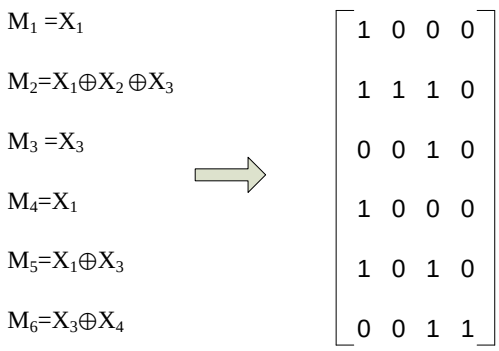


Figure 3.18 Linear Equations for MISR in Figure3.16 [15]

After the output response is shifted in to the MISR, the linear equations of MISR bits can be represented by a matrix, where each row corresponds to a MISR bit and each column corresponds to an *X*. The entry in the matrix is a "1" since *X's* present in each MISR bit corresponding to each row. Take an example in Figure 3.18, the second row corresponds to M_2 in the matrix, and three "1" located in the first three columns indicate dependence on X_1, X_2, and X_3, respectively. In order to guarantee linearly dependent for some combinations of rows, the number of *X*'s must be less than the size of the MISR because there are more rows than columns. The linearly dependent row combinations can be identified by Gauss-Jordan elimination [24], which includes performing rows operations that transform a set of columns into an identity matrix.

After Gauss-Jordan elimination, all-0 rows in the matrix have no dependence on the value of the *X*'s, as shown in Figure 3.19, where the first all-0 row is got from $M_1 \oplus M_3 \oplus M_5$. This indicates that if the MISR bits *M1*, *M3*, and *M5* are XORed together, all the *X*'s cancel out and the final value has no dependence on the *X*'s. This can be calculated as follow:

$M_1 \oplus M_3 \oplus M_5 = (X_1 \oplus O_3 \oplus O_8 \oplus O_{13}) \oplus (O_2 \oplus X_3 \oplus O_5 \oplus O_{10} \oplus O_{15}) \oplus (X_1 \oplus O_2 \oplus X_3 \oplus O_{12} \oplus O_{17}) =$

$O_3 \oplus O_5 \oplus O_8 \oplus O_{10} \oplus O_{12} \oplus O_{13} \oplus O_{15} \oplus O_{17}$

By applying XOR operation on these three bits together, the final equation depends only on non-*X* values, and each of the all-0 rows correspond to MISR bit combinations where the *X*'s cancel out. During test mode, these *X*-canceled MISR bit combinations can be compared with their fault-free values for error detection.

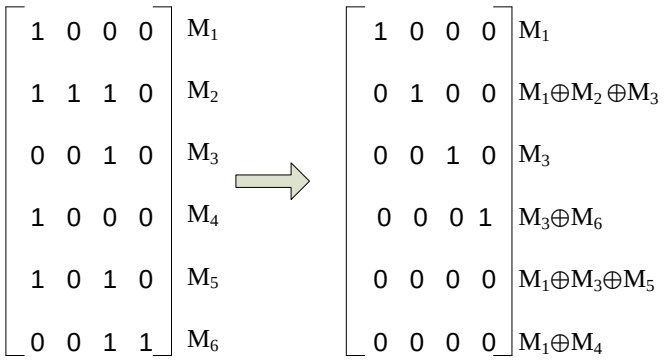


Figure 3.19 Gauss-Jordan Reduction of MISR Equations [15]

Since q *X*-canceled combinations are checked, the error coverage is approximately equal to $1-2^{-q}$. In other words, the error coverage is equivalent to using a q-bit MISR with no *X's* for signature analysis. Table 3.2 lists the error coverage for different values of q. Generally, if 7 *X*-canceled combinations are checked, then over 99% error coverage can be achieved; and if we check 13 *X*-canceled combinations, 99.99% error coverage can be obtained.

Table 3.2 Error coverage versus Number of X-canceled combination (q) [15]

X-Canceled Combination(q)	Error Coverage
1	50%
2	75%
3	87.5%
4	93.75%
5	96.88%
6	98.44%
7	99.2%
8	99.6%
9	99.8%
10	99.9%
11	99.95%
12	99.97%
13	99.99%
14	99.994%
15	99.997%
16	99.998%

3.3.2.2 *X*-Canceling MISR Architecture

The architecture of X-canceling MISR is shown in Figure 3.20. After passing through a phase-shifter, the output response is scanned into an *m*-bit conventional MISR, after that we can get the *X*-canceled combinations by using a programmable XOR network. Linear dependent combinations of MISR bits are generated by symbolic simulation. The *m*-bit selection register is performed to select a set of MISR bits which are XORed together to generate an "*X*-free" value that is compacted in the *X*-free MISR. As each *X*-free value has no dependence on *X*'s in the output responses, the final signature in the *X*-free MISR during test is deterministic and can be directly compared with the fault-free value to detect faults.

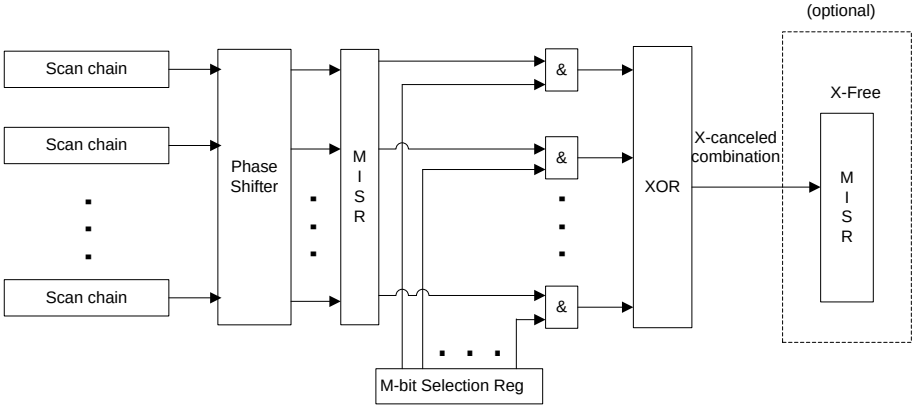


Figure 3.20 *X*-Canceling MISR Architecture [15]

As the *m*-bit MISR can compact up to *m-q X*'s with an error coverage of $1\text{-}2^{-q}$, a 256 bit MISR with a value of *q=12* can give the error coverage over 99.97%, and up to 244 (MISR bit-*q*) *X*'s can be compacted in the MISR with any number of non-*X* outputs. For the operation, the MISR can keep compacting output response data from the scan chains until the number of *X*'s reach up to 244, after that 12 *X*-canceled combinations are generated and shifted into the *X*-free MISR. Then the MISR need to be reset and can compact the output response data again.

The main advantage of *X*-canceling MISR is that the error coverage depends only on the number of *X*-canceled combinations that are checked instead of the total number of *X*'s in a scan slice. And tester storage overhead don not relate to the scan architecture, design size,

or number of test vectors, it just depends on the total number of *X*'s in the output response. Thus, this scheme is extremely efficient when the percentage of *X*'s is small. However, if the percentage of *X*'s becomes large, this scheme is not efficient due to increasing huge tester storage overhead. So it conducts a hybrid approach which combines *X*-masking logic with *X*-canceling MISR, as described in the next section.

3.3.2.3 Combining X-masking and X-canceling MISR

Since the *X*-canceling MISR scheme is not suitable for large amount of *X*'s becomes large, a hybrid method is proposed in [42] by combining *X*-masking with the *X*-canceling MISR. The basic idea of the hybrid method is to handle majority of the *X*'s that can be easily and efficiently masked, and the residual *X*'s is handled by the *X*-canceling MISR. One of main advantages of this approach is that it is very efficient to handle large amount of *X*'s in output response.

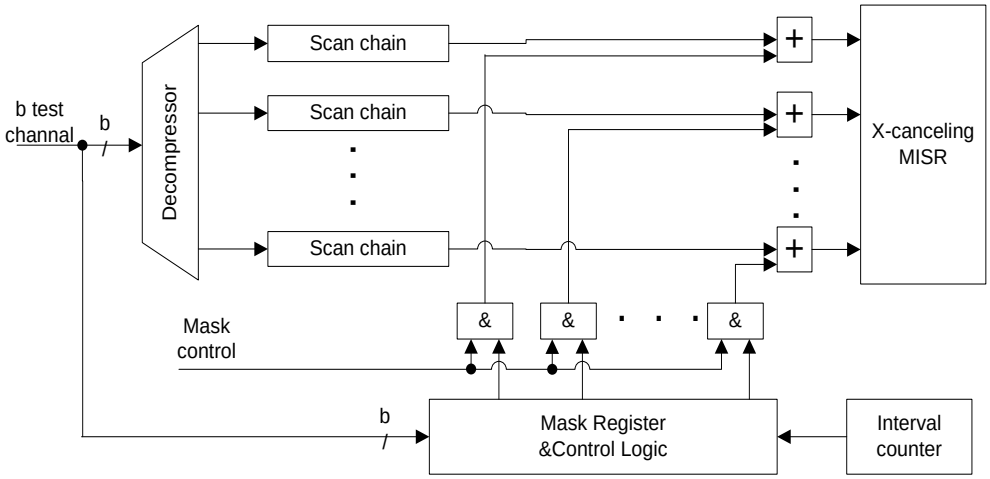


Figure 3.21 Combining X-masking and X-canceling MISR [42]

Figure 3.21 shows the architecture for *m* scan chains with a control signal, an *m*-bit masking register, an interval counter, and two logic gates per internal scan chain for a total of *2m* logic gates. The control signal is applied to determine whether or not to mask on each scan slice. Consider the case where the control signal is a '1', then the mask is applied, if the control signal is set to '0', the mask data is useless and cannot affect the scan output response data. Once

the control signal is a '1', and if a given mask data bit is set to '1', the corresponding output response data bit is masked; if mask bit is given to '0', it means no masking will occur. The goal is to consider as many adjacent scan slices as possible as well as mask as many *X*'s in those scan slices as possible without losing fault coverage. After masking, the majority of *X's* is killed and the residual small number of *X's* in the output response can be handled by *X*-canceling MISR.

Comparing the scheme of response shaper, the proposed X-canceling MISR also applies external control logic before compaction. Without using external control logic, the next section will show the X-tolerant compactor which can tolerate X's by itself.

3.3.3 X-compactor

X-compact is a typically X-tolerant test response compaction technique presented in [5]. With the X-Compact design technique, the data volume can enable up to exponential reduction in the test response and very few pins are required. For example, by applying the X-compact technique, the 500 parallel scan chains where each scan chain contains 1000 flip-flops use only 12 scan-out pins. Thus, the total overhead of test response data volume is only 90 MBs instead of 3.75 GBs. The compaction hardware has negligible area overhead and does not degrade the system performance during normal operation. Meanwhile, it guarantees the error detection and diagnosis capability of scan-based DFT even in the presence of *X's.* Another advantage is the X-Compact technique can minimum impact on current design and test flow and is independent of the fault models and test patterns. The main benefit of X-Compact technique can be list as follow: 1) test time reduction due to few pins with more scan chains; 2) test data volume reduction due to exponential reduction in the response data; 3) less the number of input/output pins; 4) low cost testers due to very few scan channels. The next two sections will show the basic algorithm for designing X-compactor with and without X's, respectively.

3.3.3.1 Compactor Design in the Absence of X's

In this section, we consider the case where none of scan chains contains X's in simulation. Figure 3.22 shows a design with eight scan chains which are directly connected to the

inputs of a compactor with eight inputs and four outputs. Consider the following two situations:

1) For a test pattern, flip-flop 4 of scan chain 1 and flip-flop 6 of scan chain 2 contain errors. The errors can be produced on pin Out1 on the fourth (sixth) clock cycle when the XOR of scan chain 1 and scan chain 2 is scanned out.
2) For a test pattern, only flip-flop 4 of scan chain 1 and scan chain 2 contain errors.

No error can be observed because the errors will cancel out when the XOR of scan chain 1 and scan chain 2on the fourth clock cycle.

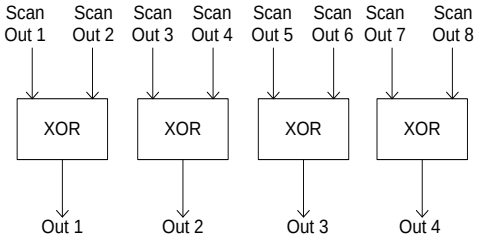


Figure 3.22 Compactor with eight inputs and outputs [5]

Figure 3.23 shows a compactor generated by the X-compact design technique with eight inputs and four outputs. This design has the following rules. When one, two, three, or a larger odd number of scan chains produce errors and no scan chain produces *X* at the same scan-out cycle, then the output produced by the compactor is guaranteed to be different from the expected compactor output at that scan-out cycle. Thus, the errors at the scan chain outputs can be detected by the compactor. This compactor designs is applied to exclusive-OR gates due to the information loss-less property of XORs.

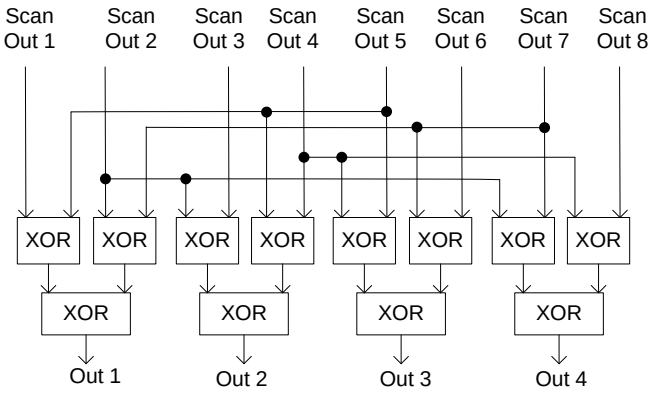


Figure 3.23 Another compactor with eight inputs and four outputs [5]

Consider a design with *n* scan chains. Then the compactor will contain *n* inputs. Suppose that the compactor *m* circuit has outputs. The compactor circuit can be represented as a binary matrix (a matrix with only 0s and 1s) with n rows and m columns. The binary matrix is called the X-compact matrix. In the matrix, each row corresponds to a scan chain and each column corresponds to a compactor output. The entry in row *i* and column *j* of the X-Compact matrix is 1 if and only if the *j*th compactor output depends on the output of the *i*th scan chain; otherwise, the entry is 0. Figure 3.24 shows the X-compact matrices corresponding to Figure3.22 and Figure3.23. The first column of the corresponding X-Compact matrix in Figure 3.24(a) has 1s in the first and the second rows because Out1 depends only on scanout1 and scanout2; The first column of the corresponding X-Compact matrix in Figure 3.24(b) has 1s in the first , second ,fifth and seventh rows because Out1 depends on scanout1, scanout2, scanout5, and scanout7.

Matrix of Figure3.22: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ (a)

Matrix of Figure3.23: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ (b)

Figure 3.24 X-compact matrices of (a) and (b) [5]

There are several theorems for systematically designing X-Compactor circuits in the absence of X's. Theorems 1–3 can be found on the theory of error-correcting codes [17].

Theorem 1: If any single scan chain produces an error at any scan-out cycle, the compactor circuit is guaranteed to produce error at that scan-out cycle if and only if no row of the X-Compact matrix contains all 0s.

Theorem 2: Errors from any one or any two scan chains at the same scan-out cycle are guaranteed to produce errors at the compactor outputs at that scan-out cycle if and only if all rows of the X-Compact matrix are nonzero and distinct.

The first two rows of the X-Compact matrix of Figure 3.22 are identical, which is against the Theorem 2.Thus, the errors from scan chains 1 and 2 at the same scan-out

cycle do not produce error on Out1. Based on Theorem 2, all rows of the X-compact matrix of Figure3.23 are distinct, simultaneous errors from any two scan chains at the same scan-out cycle are guaranteed to be detected.

Theorem 3: Errors from any one, two, or odd number of scan chains at the same scan-out cycle are guaranteed to produce errors at the compactor outputs at that scan-out cycle if every row of the X-compact matrix is nonzero, distinct, and contains an odd number of 1s.

As all rows of the X-Compact matrix are distinct and contain an odd number of 1s, the XOR of any two rows and any odd number of rows is nonzero. Thus, errors from any one, two, or odd number of scan chains at the same scan-out cycle are guaranteed to produce errors at the compactor outputs at that scan-out cycle. The matrix corresponding to compactor design in Figure 3.23 satisfies Theorem 3.

Consider X-compactor matrix with *m* columns (*m* outputs for compactor), a maximum number of distinct nonzero rows with an odd number of 1s is 2^{m-1}. To satisfy Theorem 3, the design must follow: $n \leq 2^{m-1}$, where *n* is the number of scan chains. By transforming the format to: $m \geq 1 + \log_2 n$, the relationship between scan chains(*n*) and compactors outputs (*m*) can be got , as shown in Table 3.3. It shows that 512 scan chains just need only 10 scan-out pins.

Table 3.3 Scan chains and compactor outputs: guaranteed error detection when 1, 2, or any odd number of scan chains produce errors and no scan chain produces X's at the same scan-out cycle [5]

Scan chains (n)	Compactor outputs (m)
5-8	4
9-16	5
17-32	6
33-64	7
65-128	8
129-256	9
257-512	10
513-1,024	11
1,025-2,048	12

In this section, it assumes that no scan chains produce logic values *X's* during simulation at the same scan-out cycle. However, once *X's* occur in scan chains, the error can get masked by *X's* and the defective chip is not detected. In this case, the design for X-compactor should be more sufficient in order to detect error.

3.3.3.2 Compactor Design in the Presence of X's

If a scan chain produces an *X* at a particular scan cycle, all compactor outputs that depend on this scan chain will be masked by *X* on the tester. Take an example in Figure 3.22. If flip-flop 4 of scan chain 1 records error as well as flip-flop 4 of scan chain 2 produces an *X* for a test pattern, the fourth bit of pin Out1 must be masked out by *X* for this test pattern and no error can be observed. Another example is taken in the compactor design of Figure 3.23. Assume that scan chain 1 produces an error with an X produced in scan chain 2 at the current scan-out cycle. In this case, the error on the output of scan chain 1 will not be detected as Out1 is the only compactor output that depends on the output of scan chain 1. Therefore, in order to guarantee error detection in this situation, more sufficient conditions are given in Theorem 4.

Theorem 4: An error from any scan chain and an X from any other scan chain at the same scan-out cycle are guaranteed to produce error at the compactor outputs at that scan-out cycle if and only if:

1) No row of the X-compact matrix contains all 0s;
2) For any X-compact matrix row, the submatrix obtained by removing that row and the X-compact matrix columns having 1s in that row does not contain a row with all 0s.

Figure 3.25 gives a simple example to show the compactor circuit, and the corresponding X-compact matrix is shown in Figure 3.26.

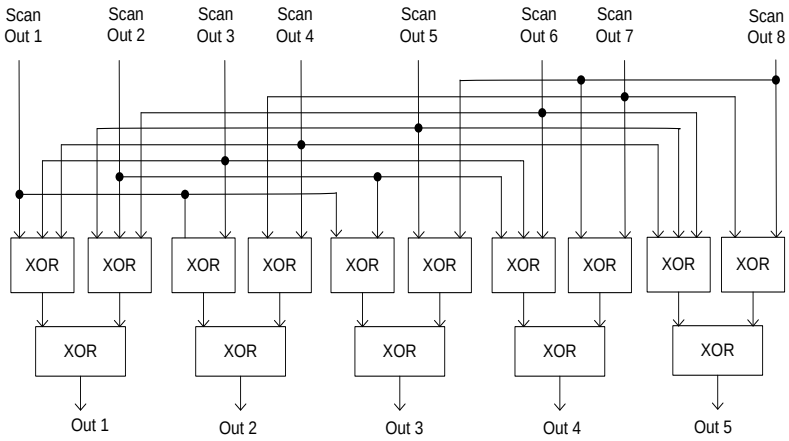


Figure 3.25 Compactor with eight inputs and five outputs with guaranteed error detection in the presence of X's [5]

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Figure 3.26 X-compact matrix of Figure3.25 compactor [5]

Suppose that scan chain 1 produces an error with an *X* produced in scan chain 2 at the same scan-out cycle. Since compactor outputs Out1, Out3, and Out4 depend on the output of scan chain 2, logic values on Out1, Out3, and Out4 will *X*. However, the error produced by scan chain 1 can be detected on compactor output Out2.

For the X-compactor matrix of Figure 3.25, if the second row and columns 1, 3, and 4, are removed, we obtain the submatrix shown in Figure 3.27. The submatrix does not have any row with all 0s. Thus, the design of X-compactor circuit satisfies Theorem 4. However, the compactor design in Figure 3.23 does not satisfy Theorem 4. This is because, after removing the second row and the first, second, and fourth columns of the X-Compact matrix, the submatrix contains all 0's in the first row.

$$\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$$

Figure 3.27 Submatrix of Figure 3.25 [5]

Corollary 1: Theorems 3 and 4 are satisfied if every row of the X-compact matrix is distinct, has an equal number of 1s, and the number of 1s in every row is odd.

The X-compact matrix of Figure also satisfies Corollary 1 because every row has only three 1s. Consider an X-compact matrix with ten columns. In order to satisfy Corollary 1, the maximum number of rows is 255 if the number of 1s in each row is five, which means the design for a compactor uses only ten outputs if it gets 252 scan chains. Table 3.4 lists the relationship between the number of scan chains and the number of compactor outputs by following Corollary 1.

Table 3.4 Scan chains and compactor outputs: guaranteed error detection when 1 scan chain produces errors and another scan chain produces X at the same scan-out cycle, or when 1, 2, or an odd number of scan chains produces errors and no scan chain produces X at the same scan-out cycle [5]

Scan chains (n)	Compactor outputs (m)
5-10	5
11-20	6
21-35	7
36-56	8
57-126	9
127-252	10
253-462	11
463-792	12
793-1,716	13
1,717-3,432	14

In order to detect errors when any one or two scan chains produce errors at the same scan-out cycle together with another scan chain producing an X at that scan-out cycle, Theorem 5 gives sufficient conditions for detecting these errors.

Theorem 5: Errors from any one or two scan chains and X from any other scan chain at the same scan-out cycle are guaranteed to produce error at the compactor outputs at that scan-out cycle if and only if:

1) No row of the X-compact matrix contains all 0s;

2) For any X-compact matrix row, all rows in the submatrix obtained by removing that row and the X-compact matrix columns having 1s in that row are distinct and nonzero.

In this case, the compactor of Figure 3.25 does not satisfy Theorem 5 because the submatrix of Figure 3.25 has identical rows. For the multiple errors and *X's*, Theorems 4 and 5 can be directly extended to Theorem 6.

Theorem 6: Errors from any k_1 or fewer scan chains and unknown logic values (*X's*) from any k_2 or fewer scan chains ($k_1+k_2 \leq n$, the total number of scan chains) at the same scan-out cycle are guaranteed to produce error at the compactor outputs at that scan-out cycle if and only if:

1) No row of the X-compact matrix contains all 0s;

2) For any set of X-compact matrix rows, any set of rows in the submatrix obtained by removing the rows in and the X-compact matrix columns having 1s in the rows in are linearly independent.

The X-Compact technique for compactor circuits is very efficient as it can significantly reduce test application time and test data volume without sacrificing test quality. Meanwhile, the X-compact technique provides a solution for handling simultaneous errors from multiple scan chains and sources of X's. Furthermore, the X-compact design technique is easy to implement as it does not require any significant change to the existing design and test flows. In next section, it shows different algorithm for designing an X-tolerant compactor.

3.3.4 i-compactor

In this section, another X-tolerant compaction scheme, i-compact, is presented in [43]. It provides an alternative solution for a compactor circuit. Comparing X-compact technique, this scheme requires comparable tester memory but fewer pins. In addition, the scheme of i-compact can provide varying degrees of error detection capability in the presence of a varying number of *X's* for the same compactor.

3.3.4.1 Design binary linear codes

The technique of i-compactor is applied to design binary linear codes. For example, a binary (*n*, *k*) includes 2^k code words with *n* bits, and *H (m=n-k)* is an $m \times n$ check matrix to represent a binary code. The valid code words is defined by the equation *HD=0*, where *D* is an $n \times 1$ vector. The test system of i-compact approach is presented in Figure 3.28, which shows the whole process for error detection in the absence of X's. For the right side, the expected n-bit test response *D* goes into the compactor, then the compactor applies the code matrix *H* to produce a check word *C = HD* of *m* bits which is considered as the compacted test response. Meanwhile, the actual test response D' and check word are produced by a chip under test on the left side of this Figure. As the actual response may contain errors, it can be represented by by adding error term *E* to the expected response *D.* An i-Compact compactor matrix with distance *d* can follow the rules:

(1) detect up to $d - 1$ single-bit errors,

(2) detect up to *e* errors in the presence of up to *x* unknowns, where $e + x < d$ or

(3) correct up to *t* errors and identify up to *x* unknowns, where $2t + x < d$.

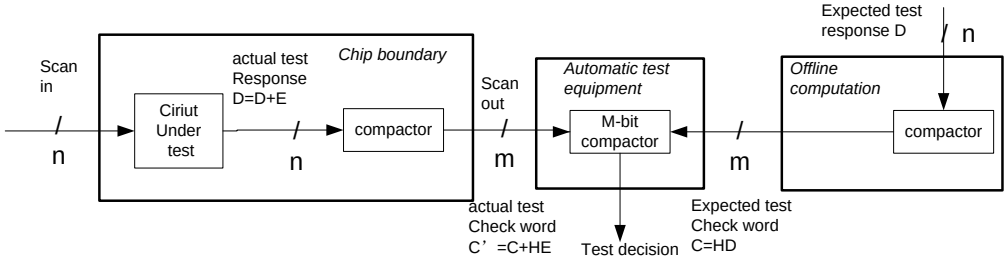


Figure 3.28 The test system of i-compact [43]

3.3.4.2 Examples

To discuss the error control capabilities of Saluja Karpovsky compactors, a series of example is shown based on the check matrix for the (7,4) Hamming Code extended the (8,4) SEC-DED code by adding a parity bit over all data bits, and the matrix is

$$H=\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 1 & 1 & 1\\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1\\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\end{bmatrix}$$

The expected test response D is represented by $d_1d_2d_3d_4d_5d_6d_7d_8$, and the expected check word C is $c_1c_2c_3c_4$. The following equation C:

$$c_1 = d_1 \oplus d_5 d_6 \oplus d_7 \oplus d_8$$

$$c_2 = d_2 \oplus d_5 \oplus d_7 \oplus d_8$$

$$c_3 = d_3 \oplus d_5 \oplus d_6 \oplus d_8$$

$$c_4 = d_1 \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_5 \oplus d_6 \oplus d_7 \oplus d_8$$

The expected check word C is pre-computed and stored in ATE memory, and the actual test response D' is also compacted by the same encoder, after that the actual check word $C' = HD'$ is sent to ATE for comparing with C. Notice that the check code H is a distance-4 code, so it can correct up to $x \leq 3$ erasures; or detect up to e errors in the presence of up to $x \leq 3 - e$ erasures. The following examples show the error control capabilities.

1) Three unknown values and no errors

Consider $D= X1X2X301101$ and $D'=01101101$. By using $C'=HD'$, we can get actual check word C' is 0101. We can also get C which depends on X_i, then C and C' need to be XORed in order to find error syndrome. In this case, the XORed result is set to 0 due to no errors, then we can solve for the X_i by the equations:

$$0 = X_1$$

$$0 = \overline{X_2}$$

$$0 = \overline{X_3}$$

$$0 = X_1 \oplus X_2 \oplus X_3$$

It gets $X_1 = 0$, $X_2 = 1$, and $X_3 = 1$ by the equation. However, when some errors are produced such as changing d_8 to 0, the error cannot be detected as the equations are not solved.

In this situation, since the number of X's and the code distance is small, all possible binary combinations of X's can be list, after that all possible check words *C* are picked to matches *C'*.

2) Two unknown values and one error

If there are two inputs including unknown values, the distance-4 code can detect any single error. Consider D=001$X_1$101$X_2$0 and D'=00101110. The actual check word is C'=HD'=0010. In order to check the expected check word C, all possible values require to be list for X_1X_2=00,01,10,11,then the corresponding values for C is 1000,0101,1001,0100,respectively. As none of these matches C', an error is checked.

3) One error and one unknown value

The code H can also be used to identify a single error in the presence of an unknown value. Consider *D* = 0100100*X*1 and *D'* = 01011001. The actual check value C' is HD'=1100, and the corresponding values of C is 0010 and 1101. If X_1=0, it gets C+C'=1110, it can match none of columns of H; but if X_1=1, it gets C+C'=0001, it can match the fourth column of H which corresponds to the bit position 4 in D', indicating that d_4 is an error.

3.3.4.3 Handling more X's

Consider the situation where the chip produces more than two X's on many scan-out cycles. In this situation, the Hamming distance-4 code can also be used instead of more expensive codes to detect errors. However, it cannot guarantee that we always detect such errors. For example, it stores 32 alternate check words in ATE in the presence of five X's. If a circuit gets 500 scan outputs and check word width is 10, then the compaction ratio is 50 and the total number of check words is 1024. The probability of a random error mapping in to the 32 stored responses is just 32/1024, which means a 96.8% detection probability can be achieved for a response with five X's. The fault coverage can be further improved by using less efficient compaction. For example, it uses more bits of check word such as 12 instead of 10, it still provides compaction factor of 40. The random error detection probability is enhanced to (1- 32/4096) = 99.2%. Similarly, more alternate check words can be stored to handle more X's. As storing multiple responses a few times out of several hundred thousand scan cycles, the storage requirement is small.

The presented i-compact is very applicable as it provide varying degrees of error detection capability in the presence of a varying number of *X's*, and it is efficient in providing a compaction factor of 50 for 500 outputs or a factor of 90 for 1000. The well-known codes for i-compact are Hamming and BCH. In the process, there are several possible approaches to trade off between test time and ATE memory requirements, and the proposed technique is also ATPG independent.

3.4 Summary

For the output response compaction, the occurrences of X's in the test responses have been the greatest barrier to improve the compaction**.** In this chapter, several relevant X-masking and X-tolerant schemes are list to handle X's and illustrate the benefit and limitation of each approach. There is a conclusion for all the presented schemes including the objective, basic concept, advantage, and disadvantage, as illustrated in Table3.5.

Table 3.5 The discussion for list schemes

Schemes		Objective	Main concept	Advantage	Disadvantage
X-masking	Conventional X-masking	Mask all the X's	Create a mask for every scan slice	Mask all the X's	Cost too much mask bit
	Reiterative X-masking	Reduce overhead of mask bit	Ensure to reuse the same mask bit	Reduce the mask bits significantly	Be efficient when just there are less number of X's
	Hierarchically configurable register	Achieve the high observability of scan cell	Apply an optimal X-masking for each scan chain	Improve the efficiency of X-masking	Require lots of time for the iteration process
X-tolerant	Response shape	Reduce the number of undetectable fault	Shift test response by the selection logic	Implement to space compactor easily	Require additional area overhead, input control data, and runtime.
	X-canceling MISR	Achieve high compression ratio	Produce X-canceled combination by control logic	Achieve high compression ratio as it gets less number of X's(less than 0.1%)	Cost too much overhead for X-mask as the number of X's increase
	X-compactor	Increase the ability for fault detection	Design space compactor by efficient algorithm	No require additional X-masking logic	Tolerate limited number of X's
	i-compactor	Increase the ability for fault detection	Design the space compactor by distance code	Increase the error detection capability	Involves postprocessing of test responses and cannot be easily implemented using current testers

CHAPTER4

THE PROPOSED SELECTIVE X-MASKING AND RESULTS

4.1 Selective X-masking

In this chapter, an X-masking logic is presented to improve the efficiency for X-masking. Comparing with previous X-masking schemes [13] [14], this scheme is more flexible and adds a few hardware overhead considered negligible. The basic idea is illustrated in this section.

As the distribution of X's has the tendency to be clustered in output responses, the majority of X's may just locate in a small number of scan chains in actual designs [8]. It is possible to mask as many X's as possible with the smallest mask data overhead. The proposed method arranges all scan chains into two groups. One group contains large amount of X's, which can be handled by the X-mask logic. Another group which consists of the small number of X's can directly go through input of the compactor without masking. The main purposes of this scheme are not only to get rid off X's, but also to decrease the mask data overhead as much as possible.

Figure 4.1 shows the architecture of proposed approach. The architecture of m scan chains contains m-bit mask register, an interval counter, m 2-to-1 MUX and m-bit selection register. The selection register is used to determine whether one scan chain can directly be sent to the compactor. For example, if the port '0'of the MUX connected with scan chain 1 is selected by selection register, the test responses of scan chain 1 can directly be sent to compactor. Otherwise, if this value is '1', the scan chain needs to be sent to the mask logic first. As the selection register is only loaded one time during whole test cycle, the overhead of this controlling data could be considered negligible.

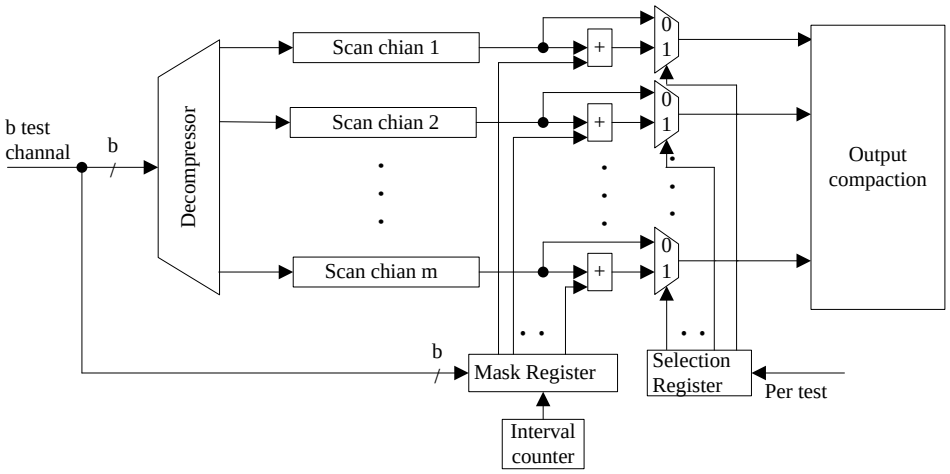


Figure 4.1 Architecture of selective X-masking for output compaction

Figure 4.1 gives the basic idea for the proposed approach. In order to understand the operation of producing the mask bit and filling the controlling data in detail, the following section describe the operations of mask bit generation, interval counter, and selection register.

4.1.1The Process of Mask Bit Generation

In this section, it shows the process of mask bit generation. In general, it includes the parts of LFSR, Phase shifter and *m*-bit mask register, as shown in Figure 4.2. In order to get mask bit for masking the X's in test output responses, the seeds coming from *b* test channels are loaded to LFSR, after that the bit sequences produced by LFSR are shifted by phase shifter, where the phase shifter is used to reduce the correlations among bit sequences through using networks of XOR gates. The mask bits generated by phase shifter are then used to mask test responses in scan chains. Notice that the total bits of mask bit generated by phase shifter are *m*, where $m > b$.

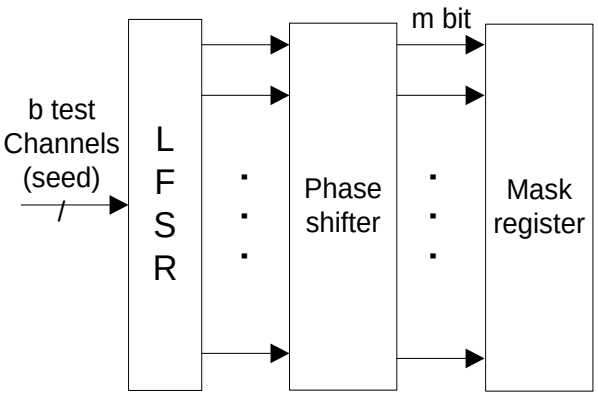


Figure 4.2 m-bit mask bit generation

Figure 4.3 shows a simple example with two test channels and a 4-stage LFSR for mask bit generation. Assume that the initial seed bits S_1-S_4 have been filled in LFSR, and the seed bits S_5-S_{10} are shifted in from two test channels. After simulation, the mask bits in mask register are represented by Z_1-Z_{12}. In the first clock cycle, the top stage of LFSR is filled by the XOR of S_2 and S_5, the second stage of LFSR is filled by S_3 XOR S_6, the third stage of LFSR is filled by the XOR of S_1 and S_4, and the bottom flip-flop is filled by S_1. After that, the value in all stage of LFSR is XORed by phase shifter. The top output of phase shifter produces the value by the XOR of the second stage of LFSR and the third stage of LFSR, the second output of phase shifter produces the value by the XOR of the second stage of LFSR and the fourth stage of LFSR, the third output of phase shifter produces the value by the XOR of the top stage of LFSR and the fourth stage of LFSR, and the bottom output of phase shifter is shifted by the fourth stage of LFSR. Finally, it gets $Z_1 = S_1 \oplus S_3 \oplus S_4 \oplus S_6$, $Z_2 = S_1 \oplus S_3 \oplus S_6$, $Z_3 = S_1 \oplus S_2 \oplus S_5$, and $Z_4 = S_1$.

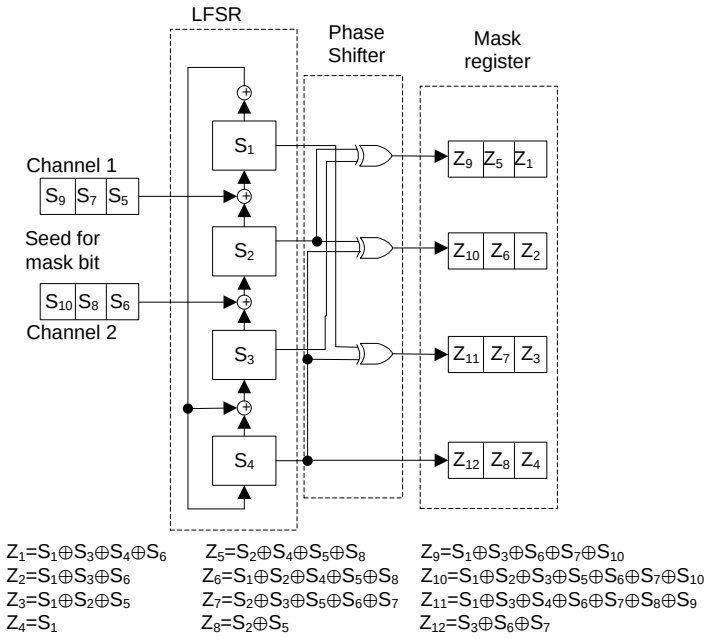


Figure 4.3 A 4-stage LFSR for mask bit generation

In the second cycle, the top stage of LFSR is filled by the XOR of the values of the second stage of LFSR ($S_3 \oplus S_6$) and S_7; the second stage of LFSR is filled by the values of the third stage of LFSR ($S_1 \oplus S_4$) and S_8; the third stage of LFSR is filled by the XOR of the values of the first stage of LFSR ($S_2 \oplus S_5$) and the fourth stage of LFSR (S_1); and the fourth stage of LFSR is filled by the XOR of the values of the first stage of LFSR ($S_2 \oplus S_5$). After the value in all stages of LFSR is XORed by phase shifter, we get $Z_5 = S_2 \oplus S_4 \oplus S_5 \oplus S_8$, $Z_6 = S_1 \oplus S_2 \oplus S_4 \oplus S_5 \oplus S_8$, $Z_7 = S_2 \oplus S_3 \oplus S_5 \oplus S_6 \oplus S_7$, and $Z_8 = S_2 \oplus S_5$. In the third cycle, the top stage of LFSR is filled by the XOR of the values of the second stage of LFSR ($S_1 \oplus S_4 \oplus S_8$) and S_9; the second stage of LFSR is filled by the values of the third stage of LFSR ($S_1 \oplus S_2 \oplus S_5$) and S_{10}; the third stage of LFSR is filled by the XOR of the values of the first stage of LFSR ($S_3 \oplus S_6 \oplus S_7$) and the fourth stage of LFSR ($S_2 \oplus S_5$); and the fourth stage of LFSR is filled by the XOR of the values of the first stage of LFSR ($S_3 \oplus S_6 \oplus S_7$). After the value in all stages of LFSR is XORed by phase shifter, it gets $Z_9 = S_1 \oplus S_3 \oplus S_6 \oplus S_7 \oplus S_{10}$, $Z_{10} = S_1 \oplus S_2 \oplus S_3 \oplus S_5 \oplus S_6 \oplus S_7 \oplus S_{10}$, $Z_{11} = S_1 \oplus S_3 \oplus S_4 \oplus S_6 \oplus S_7 \oplus S_8 \oplus S_9$, and $Z_{12} = S_3 \oplus S_6 \oplus S_7$. At the moment, the mask bits are fully generated.

4.1.2 Interval Counter

The interval counter is applied to reusing masks for multiple adjacent scan slices. It is loaded one time at the beginning of the test and controls the loading of mask register. Once the value of interval counter is set, it counts down the number of shift cycles until the value equal zero, then it is reset and returned to the initial value. As the mask bits are generated by LFSR in Figure 4.3, the mask bits are reused for three clock cycles to create a mask for test responses when the value of interval counter is set to '3', as shown in Figure 4.4.

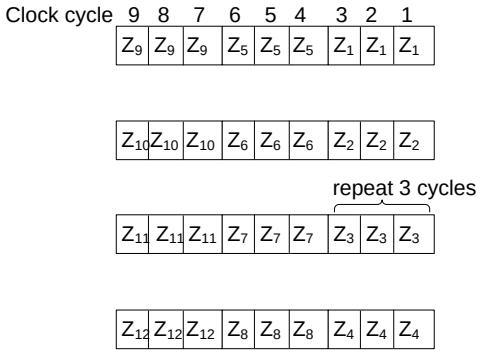


Figure 4.4 Creation of a mask for test responses (interval counter=3)

4.1.3 Selection register

The selection register is used to determine whether one scan chain requires a mask or not. The total bits of selection register are m, which is the same number of scan chains, as shown in Figure 4.5(a). Notice that the value of selection register can only be set '1'or '0'. If the value of some bit is '0', the corresponding scan chain is not necessary to mask; otherwise if the value is '1', the scan chain requires to be masked. Figure 4.5(b) gives an example with 4-bit selection register to create a mask. As the second bit and third bit of the selection register are set to'0', it is not necessary to create a mask for the corresponding second and third scan chains, so the corresponding mask bit is '?' (don't care). Since the first bit and fourth bit of the selection register are set to'1', the corresponding first and fourth scan chains are considered to be masked. As a subset of scan chains are selected to mask, less mask bits are required in comparison with a mask in Figure 4.4.

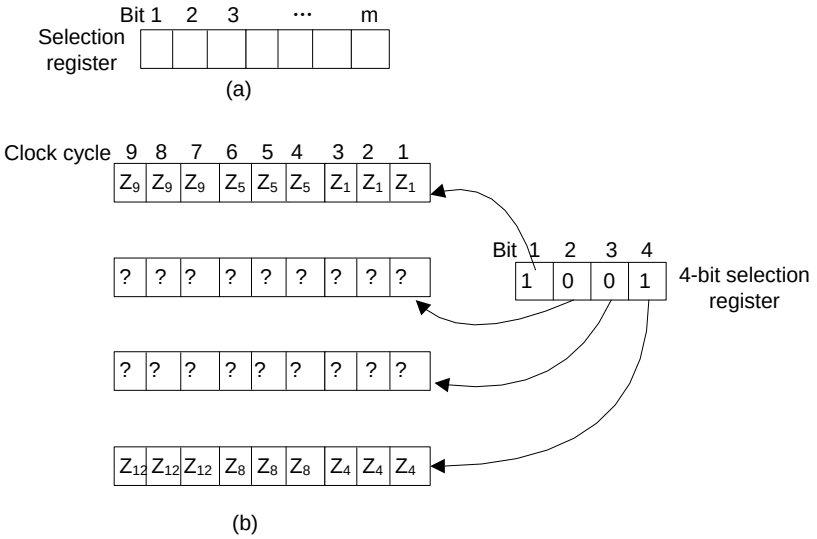


Figure 4.5 Creation of a mask for test response with selection register (interval counter=3)

As described the whole the process for mask bit generation and the principle of filling data for selection register, a simple example is given in following section to demonstrate the operation.

4.2 An Example for Selective X-masking

The basic idea is to select a subset of scan chains with most X's during the complete test for masking. The Figure 4.6(a) gives a simple example on how to select the selection register value. The example of this approach consists of only five output response patterns, each pattern includes four scan chains and five scan slices. First determine the location of all d's and X's during simulation, where d's correspond to a scan cell that needs to be observed to ensure detection of the necessary faults. As there are no X's or just less number of X's in scan chain 1, 3 and 4, the available way is to sent these scan chains directly to compactor rather than X-masking logic in these scan chains. However, the scan chain 2 contains eight X's and two d's, the density of X's in chain 2 is much lager the chain 1, 3 and 4, the scan chain 2 can be sent to the X-masking circuitry first. As a result, the bit 1, 3 and 4 in selection register is equal to '0' and the bit 2 is set to '1'in Figure 4.2(a). Meanwhile, the mask data in mask register is given in this diagram. In this example, the value of interval counter is set to '5'. For an interval of five scan slices, mask bits

can be set to ‘mask’ the pattern 3, 4, and 5 in scan chain 2 since X’s are present but d’s are not. Then, mask bits can be set to ‘not mask’ for pattern 2 in scan chain 2 since d’s are present.(‘?’ means no need to concern the given bits).The results are shown in Figure 4.6(b) wherein three X’s are left for the selective X-masking logic to handle.

Scan Chains	Output response pattern number # 5	Mask	4	Mask	3	Mask	2	Mask	1	Mask	Selection Register
1	s s s d s	?	d s s d s	?	d d s d s	?	d s s s s	?	s d d **x** s	?	0
2	**x x x** s s	1	s s s **x x**	1	s s **x** s **x**	1	s s d d **x**	0	s s s s s	?	1
3	s s s d s	?	s s s d s	?	s s d s s	?	d d s s s	?	s d s s s	?	0
4	s s s s s	?	s s s s s	?	s s s s s	?	s s s **x** s	?	s s s s s	?	0

(a)

	5	4	3	2	1
1	s s s d s	d s s d s	d d s d s	d s s s s	s d d **x** s
2	s s s s s	s s s s s	s s s s s	s s d d **x**	s s s s s
3	s s s d s	s s s d s	s s d s s	d d s s s	s d s s s
4	s s s s s	s s s s s	s s s s s	s s s **x** s	s s s s s

(b)

Figure 4.6 An example for filling selection register and mask register

The interval counter counts down the number of shift cycles until the value equal zero, then it is reset at the beginning of the scan test and a new mask data is loaded .If there are m scan chains and b tester channels, it will be take m/b clock cycles to fully load the mask data .The following section will display the approach of combining selective X-masking and X-canceling MISR and shows the process for basic operation.

4.7 Combining with X-canceling MISR

In this scheme, the presented a selective X-masking is used to handle the majority of X’s, while the X-canceling MISR is used to handle the remained X’s, as shown in Figure 4.7. The objective of this approach is to decrease the mask data overhead as much as possible. Notice that this scheme gives flexible choice between the scheme hybrid X-masking and X-canceling MISR [42] and the scheme X-canceling MISR alone [15]. If all bits of selective register are set to ‘1’, this scheme becomes the scheme hybrid X-masking and X-canceling MISR; otherwise if all bits are ‘0’, this scheme becomes the scheme X-canceling MISR. Generally, the list two cases cannot happen due to the large different distribution of X’s among scan chains. To illustrate the operation of X-masking and fault detection in X-canceling MISR, a simple will be

given as follow.

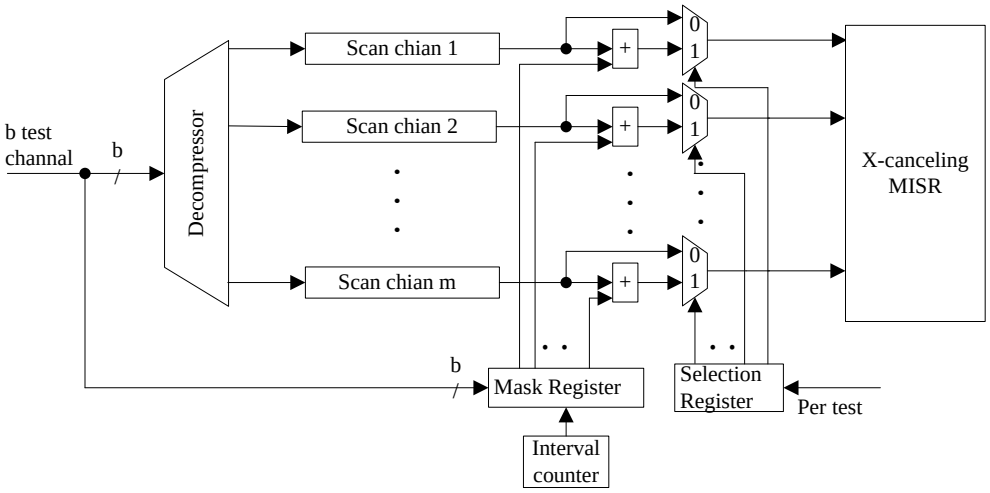


Figure 4.7 hybrid selective X-masking and X-canceling MISR

4.3.1 The Operation of the Hybrid Approach

In order to illustrate the operation of the hybrid approach, a simple example is given in Figure 4.8. It includes total eight scan chains with each five scan slices for output responses.

Scan chain	Scan slice
1	x x s s s
2	s s s s s
3	s s d x x
4	s s x x x
5	s s s s x
6	s s d s s
7	d d s s s
8	x s s s d

Figure 4.8 Output responses with eight scan chains

Assume that the interval counter in this example is set to'5'. As mentioned before, only the scan chains with large number of X's require to be masked. Thus, when the scan slices come to selective X-masking logic, the selection register only selects scan chain 1, 3, and 4 to mask due to more X's in comparison with other scan chains. After apply a mask bit for each scan chain, five X's are masked and four X's remain, as shown in Figure 4.9.

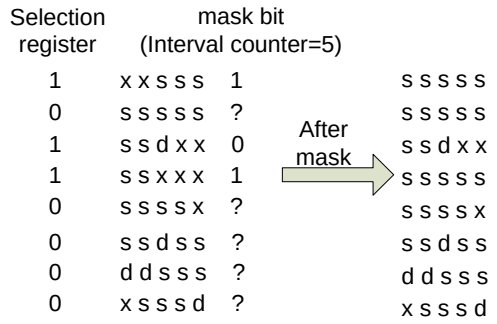


Figure 4.9 An example for selective X-masking

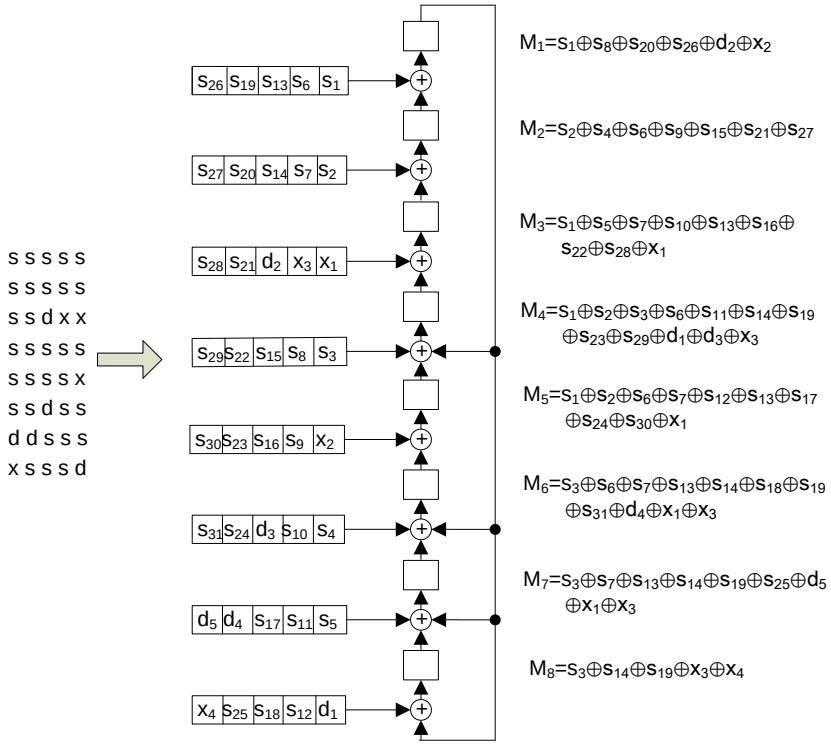


Figure 4.10 Example for symbolic simulation of MISR

After apply a mask in Figure 4.9, the scan slices are then sent to X-Canceling MISR. Figure 4.10 shows the process that the scan slices are shifted to MISR, where each scan cell is represented by a symbol. Symbol simulation is performed to obtain the final state of MISR after five clock cycles in this example. Each bit of MISR is equal to a linear combination of scan cells. For example, the final value of top bit of MISR is equal to $s_1 \oplus s_8 \oplus s_{20} \oplus s_{26} \oplus d_2 \oplus X_2$.

Assume that each symbol X_i has an X value and each symbol d_i and s_i has a

non-*X* value. Moreover, assume each symbol d_i corresponds to a scan cell that needs to be observed to ensure detection of the necessary faults for this particular test vector. Only the X dependence is considered and all non-*X* values are observed.

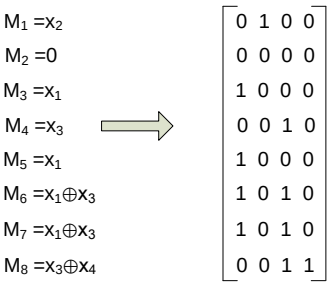


Figure 4.11 Linear equation for Figure 4.10

The linear equations for each MISR bit can be represented as a matrix, where each row corresponds to a MISR bit and each column corresponds to an X. Notice that each entry in the matrix is a 1 if the MISR bit corresponding to the row depends on the X corresponding to the column, as shown in Figure 4.11. For example, the sixth row of the matrix corresponds to M_6, and the 1's in the first and third columns indicate dependence on X_1 and X_3. If the number of X's is less than the size of the MISR, then there are more rows than columns. Thus, some combinations of rows are guaranteed to be linearly dependent.

Gauss-Jordan elimination [24] is used to determine the linearly dependent row combinations, and the operation is to transform a set of columns into an identity matrix. Figure 4.12 shows the matrix in Figure 4.11 after Gauss-Jordan elimination has been performed. We can get the all-0 rows in the matrix after Gauss- Jordan elimination, where the all-0 row is that it has no dependence on the value of the X's. For example, the combination rows are XORed together to get the all-0 row such as $M_3 \oplus M_5$. In this matrix, it gets total four all-0 rows,which can be used to compare with fault-free values for fault detection as the corresponding MISR bit combinations get all X's cancelled out.

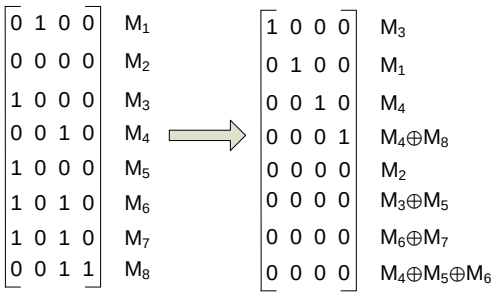


Figure 4.12 Gauss- Jordan elimination

After apply a mask in Figure 4.9, the scan slices are not only sent to X-Canceling MISR, but also it can be sent to X-tolerant compactor for compaction, as shown in next section.

4.8 The X-tolerant Compactor with Selective X-masking

In this scheme, we apply X-tolerant compactor with selective X-masking logic, as shown in Figure 4.13. The selective X-masking is used to handle the majority of X's, while the remained X's is handled by X-tolerant compactor. The main objective of this approach is to improve the observability of the test responses.

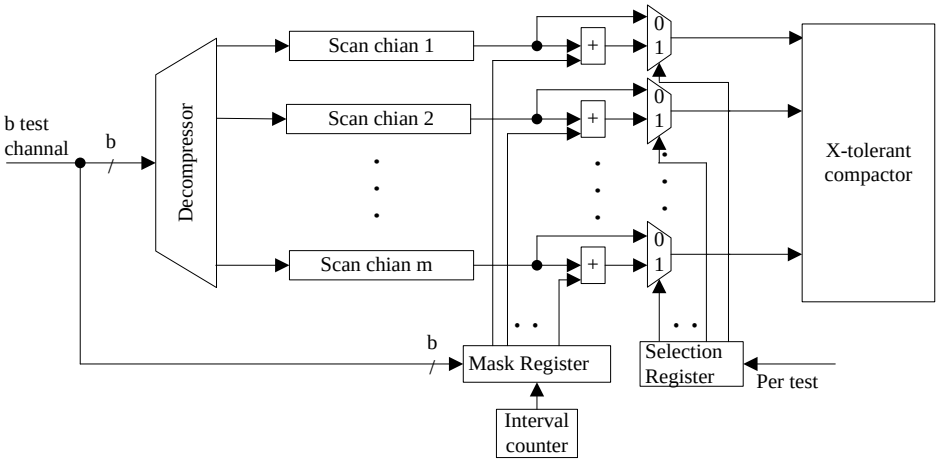


Figure 4.13 The X-tolerant compactor with selective X-masking logic

In order to illustrate the approach of the selective X-masking for X-tolerant compactor, it takes a simple X- compactor with eight inputs and five outputs, and the compactor circuit is shown in Figure 4.14 .The corresponding X-compact matrix is shown in Figure 3.15.

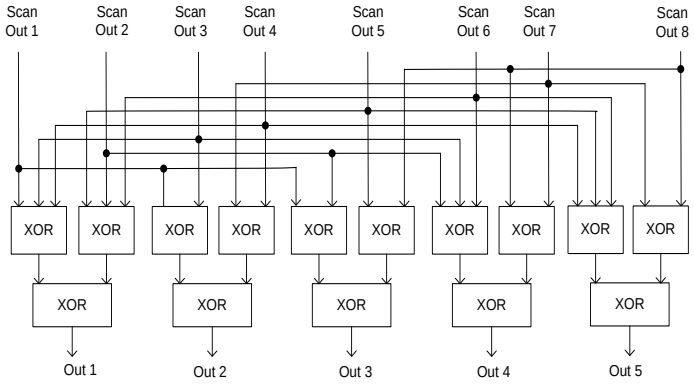


Figure 4.14 Compactor with eight inputs and five outputs

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Figure 4.15 X-compact matrix for Figure 4.14 compactor

After apply a mask in Figure 4.9, the scan slices can also be sent to the X-compactor for compaction. The operation is shown in Figure 4.16. The masked scan slices are sent to the input of X-compactor, where d_i corresponds specified bit that needs to be observed to detect faults and bit s is don't care bit. After five clock cycle, the final values are propagated to the output of the compactor, which can be used to compare with fault-free values for fault detection. Finally, all the specified bits (d_1-d_5) can be observed with the presence of four X's.

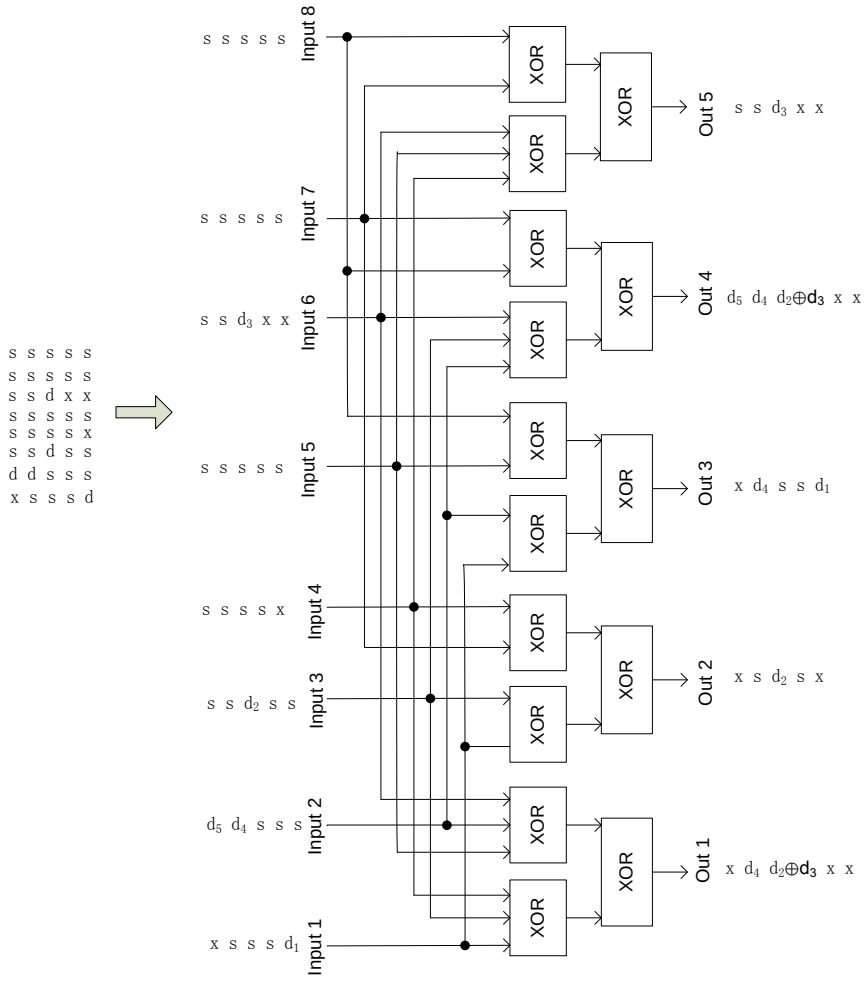


Figure 4.16 Example for compaction with X-compactor

4.5 Results

Experiments are performed on four ISCAS89 benchmark circuits to implement the proposed method. In Table 4.1, the first column lists the circuit used in this experiment. The second column gives the total number of the scan elements. The third column lists the number of scan chains. The last column shows the number of test patterns.

Table 4.1 ISCAS 89 benchmark circuit

Circuit	Scan element	# of chain	# of test pattern
s15850	534	107	229
s13207	638	126	343
s38417	1636	164	485
s38584	1426	143	464

For the scheme in Figure 4.7, experiments were performed using proposed method with four ISCAS 89 benchmark circuits. The results are shown in Table 4.2. The 0.5% of X's are randomly assigned in the test responses-specifically, 90% of X's from 10% of the scan chains. The first column shows the benchmark circuits. The remaining columns give the comparison of the compression ratio for X-canceling, X-masking and X-canceling, and proposed method, respectively. It can easily see that the proposed method provides the significant reduction of test data overhead in all cases.

Table 4.2 compression ratio with 0.5% X's

Circuit	X-canceling MISR alone	X-masking& X-canceling	Proposed method
s15850	19.8x	10.5x	**35.4x**
s13207	24.2x	30.1x	**69.8x**
s38584	23.1x	14.8x	**39.2x**
s38417	23.2x	26.7x	**66.9x**

By observing the positions of both d's and X's in a set of test patterns, the values of the selection register and mask data can be calculated for dividing scan chains into two groups: a large number of X's and few X's. The X-masking logic is inserted between the output of scan chains and the input of X-canceling MISR to handle the group containing a large number of X's. The group with few X's can be handled directly by the X-canceling MISR. The experimental results were shown that the proposed method provides significantly improvement on the amount of compression. By inspecting the distribution of X's, the proposed method will give more effective when the majority of X's is located in small parts of scan chains.

For the scheme in Figure 4.13, it reports the results to show how the selective X-masking logic can improve the percentage of observability for output responses (%of Obs.

response) for the X-compactor (one of X-tolerant compactors). In this experiment, 1% of X's are randomly generated in the test responses-specifically, 90% of X's filled in 10% of the scan chains.

In Table 4.3, in the second column it shows the percentage of X's masked in the test cubes by the selective X-masking logic (S. mask). The total overhead of mask bits is shown in the third column. The fourth column (or fifth column) shows the percentage of observability for output responses with (or without) selective X-masking logic. The sixth column is the improvement of observability for output responses. By comparing the fourth and fifth columns, the percentage of observability for output responses is improved in the range of 7.93% to 14.32% (11.67% on the average for four circuits).The results indicate that the selective X-masking logic can achieve a significant improvement of observability for output responses in the comparison.

Table 4.3 Comparison between with and without using selective X-masking with 1% X's

Circuit	X-compactor				
	With S. mask			W/o S. mask	Improve-ment% (a-b)/a
	%X's mask	Total Mask bits	% of Obs. Response(a)	% of Obs. Response(b)	
s15850	64.8	1,511	**96.33**	82.53	14.32
s13207	80.9	2,130	**98.85**	85.29	13.73
s38417	83.6	7,658	**99.63**	88.98	10.69
s38584	65.2	8,024	**97.71**	89.96	7.93
Avg.	73.6		**98.13**	84.60	11.67

When the number of X's has been increased to 2%, the percentage of observability was enhanced by 34.11% on average for four circuits as shown in Table 4.4.

Table 4.4 Comparison between with and without using selective X-masking with 2% X's

Circuit	X-compactor				
	With S. mask			W/o S. mask	Improve-ment% (a-b)/a
	%X's mask	Total Mask bits	% of Obs. Response(a)	% of Obs. Response(b)	
s15850	63.1	1,854	**89.05**	56.22	36.87
s13207	81.6	3,187	**96.80**	61.55	36.42
s38417	83.9	11,171	**98.58**	69.34	29.66
s38584	64.8	10,316	**92.20**	61.34	33.47
Avg.	73.4		**94.16**	62.11	34.11

In Tables 4.5 and Tables 4.6, it shows the results comparing the overhead of mask bits between our approach and reiterative X-masking [14] in the presence of 1% and 2% of X's. As the results show, the reduction of mask bits is 77.67% and 72.20% on average, respectively.

Table 4.5 The overhead of mask bits based on 1% of X's

Circuit	Mask bits		Reduction %(b-a)/b
	Our approach (a)	Reiterative X-masking [14] (b)	
s15850	1,511	9,532	84. 14
s13207	2,310	7,937	73.16
s38417	7,658	29,732	74.32
s38584	8,024	38,303	79.05

Table 4.6 The overhead of mask bits based on 2% of X's

Circuit	Mask bits		Reduction %(b-a)/b
	Our approach (a)	Reiterative X-masking [14] (b)	
s15850	1,854	9,975	81.41
s13207	3,184	9,199	65.38
s38417	11,171	33,964	67.10
s38584	10,316	41,122	74.91

The experimental results were shown that the proposed method can achieve significant improvement of observability for scan cells and reduce the X-masking data overhead significantly. In addition, the proposed method is effective as the majority of X's is located in small parts of scan chains.

4.6 Summary

In this section, the general idea of a selective X-masking is firstly presented. Then, it combines the selective X-masking logic with X-canceling MISR and X-tolerant compactor, respectively. To show the performance, the results are implemented with using the proposed masking logic and without using the proposed masking logic. For the experimental results, the presented X-masking logic can achieve 53.94% improvement of the compression ratio on average based on 0.5% X's for X-canceling MISR and can improve obviously the observability of scan cells (34.11% improvement on average based on 2% X's and 11.67% improvement on average based on 1% X's) for X-tolerant compactor.

CHAPTER 5

CONCLUSION AND DISCUSSION

5.1 Conclusion

For the output response compaction, an efficient X-masking scheme requires to be developed to handle X's appeared in the test responses. The good performance for an X-masking logic is that the masking logic can handle more number of X's with smaller mask bits without losing fault coverage.

Since the X-canceling MISR is extremely efficient only when the percentage of X's is very small (less than 0.1%) while X-tolerant compactor can just tolerant limited number of X's, As the number of X's increase in test responses, the present scheme of X-canceling MISR and X-tolerant compactor are not efficient anymore. In this case, this work adds the additional X-masking logic in front of either X-canceling MISR or X-tolerant compactor. As the majority of X's may just locate in a small number of scan chains in actual design, the presented selective X-masking logic is applied to mask a subset of scan chains with large density of X's to improve efficiency of X-masking, and the few remaining small number of X's can be handled by either X-canceling MISR or X-tolerant compactor.

For the implementation, it mainly computes the overhead of mask data and the observability of scan cells for compactor with the presented X-masking logic. In experimental results, the selective masking logic can give a better performance compared with the previous X-masking schemes [13] [14].

5.2 Discussion

The proposed selective X-masking gets lots of advantages based on the large different distribution of X's in scan chains. However, it gets several limitations discussed as follow.

5.2.1 Advantages

1) The presented selective X-masking logic can be easily integrated in digital circuits as the additional selection register requires only one 1-bit wide test channel.
2) By inspecting the distribution of X's, the proposed method is extremely efficient when most X's is only distributed in a small subset of scan chains.
3) Comparing with several X-masking schemes [13] [14], the presented method is more flexible because the partial scan chains are masked.

5.2.2 Limitations

1) When the percentage of X's is very small, the either scheme of X-canceling MISR or X-tolerant compactor is very efficient. Then, it is not necessary to combine with either X-canceling MISR or X-tolerant compactor as the proposed approach.
2) The presented approach is very efficient only if the majority of X's locate in a small number of scan chains (no more than 10% of all scan chains). However, the distribution of X's among scan chains might be large different in actual circuits.

5.2.3 The future work

1) Another type of benchmark circuits may be selected for the further implementation based on the proposed X-masking logic. Notice that the four ISCAS89 benchmark circuit are performed based on large number of scan cells and complexity.
2) Another efficient X-masking logic requires to be developed for the further research.

REFERENCES

[1] P. H. Bardell and W. H. McAnney, Self-testing of multiple logic modules, in *Proc. Int. Test Conf., pp. 200–204, 1982.*

[2] L. H. Goldstein, Controllability/observability analysis of digital circuits, *IEEE Trans.Circuits Syst, pp.685–693, 1979.*

[3] E. J. McCluskey, *Logic Design Principles: With Emphasis on Testable Semiconductor Circuits*, Prentice Hall, Englewood Cliffs, NJ, 1986.

[4] M.C.-T.Chao,S.wang, and S.T.Chakredhar, K.T.Cheng "Response Shaper:A Novel Technique To Enhance Unknown Tolerance for Output Response compaction," *IEEETrans. On Computer-Aided Design, pp. 80-87, Nov. 2005.*

[5] Mitra, S., and K.S. Kim, "X-Compact: An Efficient ResponseCompaction Scheme," *IEEETrans. on Computer-Aided Design, Vol. 23,No. 3,pp. 421-432, Mar. 2004.*

[6] K. K. Saluja and M. Karpovsky, "Testing computer hardware through data compression in space and time," *in Proc. ITC, pp. 83-88, 1983.*

[7] Rajski, J., J. Tyszer, C. Wang, and S.M. Reddy, "Finite Memory Test Response Compactors for Embedded Test Applications," *IEEE Trans, pp. 622-634,2005.*

[8] J. Rajski et al, "Convolutional compaction of test responses," *in Proc.ITC, pp. 745-754, 2003.*

[9] L.T. Wang, C.-W. Wu, X. Wen, VLSI Test Principles and Architectures, *Morgan Kaufmann, 2006.*

[10] V. Chickermane, B. Foutz, and B. Keller, Channel Masking Synthesis for Efficient On-Chip Test Compression, *IEEE International Test Conference,pp.452-460, 2004.*

[11] Barnhart, C., V. Brunkhorst, F. Distler, O.Farnsworth, B. Keller, and B. Koenemann**,** "OPMISR: the Foundation for Compressed ATPG Vectors," *Proc. of International Test Conference, pp.748-757, 2001.*

[12] Pomeranz, I., S. Kundu, and S.M. Reddy, "On Output Response Compression in the Presence of Unknown Output Values," *Proc. Of Design Automation Conference, pp. 255-258, 2002.*

[13] Naruse, M., I. Pomeranz, S.M. Reddy, and S. Kundu, "On-Chip Compression of Output Responses with Unknown Values Using LFSR Reseeding," *Proc. of International Test Conference, pp. 1060-1068, 2003.*

[14] Richard putmen, "Using reiterative LFSR based X-masking to increase output compression in presence of unknowns**"** *GLSVLSI'08, pp.355-358, 2008.*

[15] Touba, N.A., "X-Canceling MISR – An X-Tolerant Methodology for Compacting Output Responses with Unknowns Using a MISR," *Proc. of International Test Conference, Paper 6.2, 2007.*

[16] S. Mitra, and K. S. Kim,"X-Compact: an efficient response compaction technique for test cost reduction", *IEEE International Test Conference, pp.311-320, 2002.*

[17] P. Wohl, J. A. Waicukauski, S. Patel, and M. B. Amin, "X Tolerant Compression and Application of Scan-ATPG Patternsin a BIST Architecture", *IEEE International Test Conference,pp.727-736, 2003.*

[18] N. K. Jha and S. K. Gupta, *Testing of Digital Systems*, Cambridge University Press,Cambridge, U.K., 2003.

[19] E. B. Eichelberger and T. W. Williams, A logic design structure for LSI testability, in *Proc. Des. Automat. Conf., pp. 462–468, 1977.*

[20] E. B. Eichelberger and T. W. Williams, A logic design structure for LSI testability, *J. Des. Automat. Fault-Tolerant Comput.*, 2(2), *pp.165–178, 1978.*

[21] S. DasGupta, P. Goel, R. G. Walter, and T. W. Williams, A variation of LSSD and its implications on design and test pattern generation in VLSI, in *Proc. Int. Test Conf., pp. 63–66 1982.*

[22] C. E. Stroud, *A Designer's Guide to Built-In Self-Test*, Springer Science, Boston, MA, 2002.

[23] S. M. Reddy, K. Miyase, S. Kajihara, and I. Pomeranz, On test data volume reduction for multiple scan chain designs, in *Proc. IEEE VLSI Test Symp., pp. 103–108, 2002.*

[24] C. G. Cullen, *Linear Algebra with Applications*, Addison-Wesley, Boston, MA, 1997.

[25] C. V. Krishna, A. Jas, and N. A. Touba, Test vector encoding using partial LFSR reseeding, in *Proc. IEEE Int. Test Conf., pp. 885–893, 2001.*

[26] B. K ̈onemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, and D. Wheater, A SmartBIST variant with guaranteed encoding, in *Proc. Asian Test Symp., pp. 325–3302001*.

[27] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherje, Embedded deterministic test, *IEEE Trans. Comput.-Aided Des.*, 23(5), *pp.776–792, 2004*.

[28] K.-J. Lee, J. J. Chen, and C. H. Huang, Using a single input to support multiple scan chains, in *Proc. Int. Conf. on Computer-Aided Design, pp. 74–78. 1998.*

[29] K.-J. Lee, J. J. Chen, and C. H. Huang, Broadcasting test patterns to multiple circuits, *IEEE Trans. Comput.-Aided Des.*, 18(12), *pp.1793–1802, 1999.*

[30] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, A. Ferko, B. Keller, D. Scott, B. K ̈onemann, and T. Onodera, Extending OPMISR beyond 10 × scan test efficiency, *IEEE Des. Test Comput.*, 19(5), *pp.65–73, 2002.*

[31] J. Rajski, J. Tyszer, C. Wang, and S. M. Reddy, Finite memory test response compactors for embedded test applications, *IEEE Trans. Comput.-Aided Des.*, 24(4), *pp.622–634, 2005.*

[32] Y. Han, Y. Xu, A. Chandra, H. Li, and X. Li, Test resource partitioning based on efficient response compaction for test time and tester channels reduction, in *Proc. Asian Test Symp., pp. 440–445, 2003.*

[33] Y. Han, Y. Hu, H. Li, and X. Li, Theoretic analysis and enhanced X-tolerance of test response compact based on convolutional code, in *Proc. Asia and South Pacific Design Automation Conf., pp. 53–58 2005.*

[34] Y. Han, X. Li, H. Li, and A. Chandra, Test resource partitioning based on efficient response compaction for test time and tester channels reduction, *J. Comput. Sci. Technol.*, 20(2), *pp.201–210, 2005.*

[35] E. J. McCluskey, D. Burek, B. Koenemann,S. Mitra, J. Patel, J. Rajski, J. Waicukauski, "Test Data Compression," IEEE Design & Test of Computers, vol. 20, *pp. 76 - 87, 2003.*

[36] H. Tang, C. Wang, J. Rajski, S. M. Reddy, J. Tyszer and I. Pomeranz, "On efficient X-handling using a Selective Compaction Scheme to Achieve High Test Response Compaction Ratios," in Proc. International Conference on VLSI Design, *pp. 59–64, 2005.*

[37] P. Wohl, J. A. Waicukauski, and S. Patel, Scalable selector architecture for X-tolerant

deterministic BIST, in *Proc. Design Automation Conf., pp. 934–939,2004.*

[38] Y. Han, Y. Hu, H. Li, and X. Li, Theoretic analysis and enhanced X-tolerance of test response compact based on convolutional code, in *Proc. Asia and South Pacific Design Automation Conf, pp. 53–58, 2005*.

[39] E. H. Volkerink and S. Mitra, Response compaction with any number of unknowns using a new LFSR architecture, in *Proc. Design Automation Conf., pp. 117–122,2005*.

[40] J. Rajski, J. Tyszer, C. Wang, and S. M. Reddy, Finite memory test response compactors for embedded test applications, *IEEE Trans. Comput.-Aided Des.*, 24(4), *pp.622–634, 2005*.

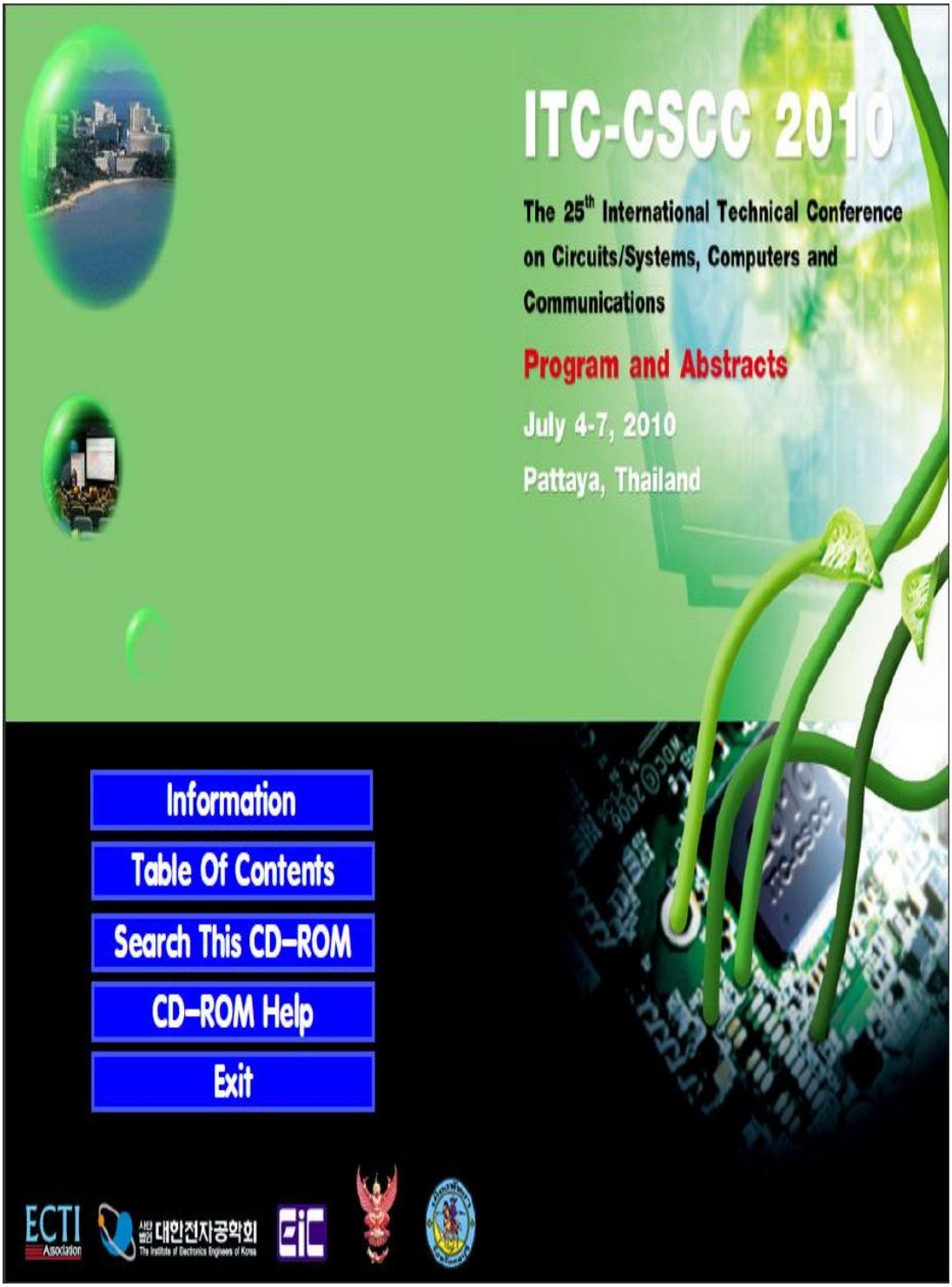
[41] T. Rabenalt, M. Goessel, A. Leininger, "Masking of X-values by Use of a Hierarchically Configurable Register" 14th IEEE European Test Symposium, *pp.149-154, 2009*.

[42] Ritesh Garg, Richard Putman, Nur A. Touba, "Increasing Output Compaction in Presence of Unknowns Using an X-Canceling MISR with Deterministic Observation," vts, 26th IEEE VLSI Test Symposium (vts 2008), *pp.35-42,2008*.

[43] Janak H. Patel, Steven S. Lumetta, Sudhakar M. Reddy, "Application of Saluja-Karpovsky Compactors to Test Responses with Many Unknowns," vts, IEEE VLSI Test Symposium, *pp.107-121, 2003*.

APPENDIX

Appendix
Published papers


ITC-CSCC 2010
The 25th International Technical Conference on Circuits/Systems, Computers and Communications
Program and Abstracts
July 4-7, 2010
Pattaya, Thailand
Information
Table Of Contents
Search This CD-ROM
CD-ROM Help
Exit
ECTI Association
대한전자공학회
The Institute of Electronics Engineers of Korea
EIC

Hybrid Selective X-masking and X-canceling MISR for Test Responses

Xu Shubing and Asst. Prof. Dr. Taweesak Reungpeerakul
Department of Computer Engineering, Faculty of Engineering, Prince of Songkla University, Thailand
5110120131@email.psu.ac.th, rtaweesak@coe.psu.ac.th

ABSTRACT

This paper presents a hybrid technique in order to handle unknown values (X's) in the test responses. It combines the main advantages of selective X-masking and X-canceling multiple input shift register (MISR). The selective X-masking is used to handle high density of X's in scan chains, while the X-canceling MISR is applied to tolerate the remained X's. The experimental results for ISCAS89 benchmark circuits have indicated the significant improvement in terms of test data overhead.

Key words-X-masking, Test Response Compaction, MISR

I.INTRODUCTION

In order to reduce the test data volume as well as the number of required channels on the ATE (automatic test equipment), a new solution needs to be concerned for both input stimulus compression and output response compaction. For the output response compaction, the presence of unknown values (X's) in the good-circuit test responses has been the greatest barrier to effective the compaction. If there is no unknowns in the good-circuit responses, a time compactor, such as Multiple Input Signature Registers (MISR), can compact an infinitely long output sequence into a fixed-length signature [1]. However, when X's appear in the responses, they can cause an unpredictable signature, from which no faulty-circuit signature could be distinguished. One of the major issues for test compaction is how to handle test responses containing X's. The sources of X's are caused by several conditions such as bus contention, uninitiated memory, un-modeled logic, floating tri-states bus, etc.

Basically there are three types of compactors: space compactors [2][3], time compactors [3] and finite memory compactors [4][5].When X's are introduced to space compactors, test responses in the current clock cycle going through the compactor may be corrupted. The time compactor, for example the MISR, is intolerance of unknown values. Finite memory compactors have to trade off in terms of the compaction ratio versus the number of X's.

Lots of schemes have been proposed to handle X's in the output response. There are three techniques to handle X's: X-blocking [6], X-masking [7][8][9][10] and X-tolerant [2][4][11][12][13]. The X-blocking scheme needs extra logic in the circuit-under-test (CUT) [1], it may result in fault coverage lost and additional area overhead. The X-masking techniques are widely proposed in previous work [7] [8] [9]. In one of X-masking techniques, the conventional LFSR X-masking scheme guarantees to mask all X's and keeps specified bits (d's), where the d's contain values require to be observed for the detection of one or more faults. However, a large amount of mask data is generated for masking every scan slice. Another X-masking scheme is called variable reiterative LFSR X-masking [10], the volume of mask data decreases greatly due to reusing the same masking bit until the conflict of the mask bit position happens. It is necessary to use interval counter for controlling. However, this scheme is efficient only a limited small number of X's.

Instead of masking the X's, X-tolerant schemes have been widely introduced in [2][12][13]. In the X-compactor [2], XOR gates are used to minimize the impact of non-X value being masked by X's. In the schemes of [12] [13], there are N scan chains and the compacted test responses output are M bits on the tester, where N>M. The test responses are propagated from scan chains with non-X values and X's to different compactor outputs. The X's are corrupted at the compactor outputs. However, the compaction ratio is extremely degraded as the increasing X's and corrupted outputs may decrease fault coverage. Another X-tolerant scheme called X-canceling MISR [11] has been proposed. However, the X-canceling MISR scheme is effective only when the percentage of X's is especially small.

In this paper, we proposed the hybrid scheme which consists of selective X-masking and X-canceling MISR. The benefit of this hybrid approach is to increase efficiency of masking X's by selective X-masking logic, which just mask

the scan chains with high density of X's, after that the rest of the X's go through to the X-canceling MISR. The experimental result shows that, the output compression ratio can evidently increase when majority of X's are located in a small number of scan chains

II.OVERVIEW OF X-CANCELING MISR

This section gives an overview of an X-canceling MISR, the architecture of an X-Canceling MISR has been proposed in [11]. The main idea of this work is the X is represented by a unique symbol, when the output streams are scanned into MISR, each linear combination is produced by each bit of MISR signature .Since the symbols corresponding to the X's are propagated into MISR bits, those combinations are linearly dependent on the symbols. In order to get some combinations that have no dependence on the value of X's, it separates the non-X values from the X's using Gauss-Jordon elimination. After Gauss-Jordon elimination, the values of some rows have no X's called all-0 rows and the all-0 rows combination is named as X-canceled combination. The error coverage is only determined by the number of X-canceled combinations. Only limited X-canceled combinations are required, for example, once q X-canceled combinations are checked, the error coverage will be approximately equal to $1-2^{-q}$, Such as, if 10 X-canceled combinations are checked, over 99.9% error coverage is obtained.

As the total control data just depends on total number of X's in output response ,the main advantage of this scheme is that the large compression ratio can be achieved when the percentage of X's is especially low in test cube. Another advantage is that error coverage just depends on the number of X-canceled combinations rather than the number of X's in scan slice. However, the X-canceling MISR scheme is not effective when the density of X's is high.

III SELECTIVE X-MASKING AND X-CANCELING MISR

The test responses have been observed that the occurrences of X's have the tendency to be clustered, which means high percentage of X's is distributed in a small number of scan chains [5]. It is possible to mask as many X's as possible with the smallest mask data overhead. The proposed method arranges scan chains into two groups. One group contains large amount of X's handled by reiterative X-mask logic [10]. Another group consists of the rest of X's handled directly by X-canceling MISR [11] without using the mask logic. Fig.1 shows the architecture of proposed approach. The main purposes of this scheme are not only to get rid off X's, but also to decrease the mask data overhead without losing fault coverage.

In Fig. 1, the architecture of m scan chains contains m-bit mask register, an interval counter, m 2-to-1 Multiplexers (MUX) and m-bit selection register. The selection register is used to determine whether X's directly sent to X-canceling MISR. For example, if the port 0 of the top MUX is selected by selection register, the top scan chain can directly be sent to X-canceling MISR, otherwise the scan chain needs to be sent to the mask logic before passing through the X-canceling MISR. In other words, if all bits of selection register are zeros, this architecture becomes X-canceling MISR; otherwise it combines X-masking and X-canceling MISR. Because the selection register is loaded one time per test pattern, the mask data overhead is negligible.

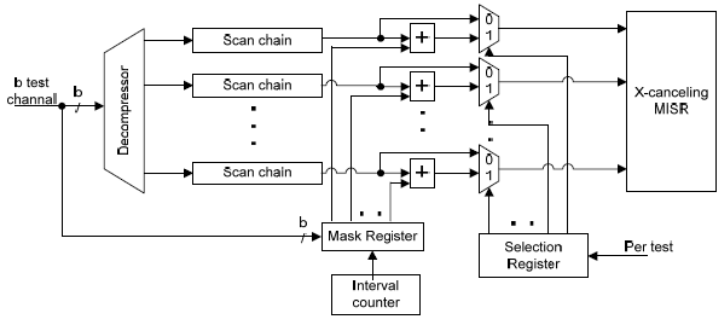


Figure1: Hybrid selective X-masking and X-Canceling MISR

Fig.2 demonstrates an example on how to assign the values of the selection register. There are five output response patterns. Each pattern contains four scan chains and five scan slices. Since there are few numbers of X's in scan chain 1, 3,and 4, then these scan chains can be connected directly to X-canceling MISR rather than X-masking logic. However, the scan chain 2 contains eight X's and two D's. The density of X's in chain 2 is much lager than in the chain 1, 3, and 4, thus the scan chain 2 is connected to the reiterative X-masking. As a result, the data bit 1, 3 and 4 of selection register equal to '0' and the data bit 2 is set to '1'. For each scan slice, mask data of mask register is shown in Fig. 2a. For an interval of five scan slices, mask bits set to '1' associated to the pattern 3, 4, and 5 in scan chain 2 since X's are presented with no d's. However, mask bit of pattern 2 in scan chain 2 is '0' because of the presence of d's. ('?' bits are don't care data). The result after passing through the reiterative X-masking is shown in Fig.2b. There are three X's left which can be handled by the X-canceling MISR.

Scan Chains	Output response pattern number # 5	Mask	4	Mask	3	Mask	2	Mask	1	Mask	Selection Register
1	sssds	?	dssds	?	ddsds	?	dssss	?	sddxs	?	0
2	xxxss	1	sssxx	1	ssxsx	1	ssddx	0	sssss	?	1
3	sssds	?	sssds	?	ssdss	?	ddsss	?	sdsss	?	0
4	sssss	?	sssss	?	sssss	?	sssxs	?	sssss	?	0

(a)

	5	4	3	2	1
1	sssds	dssds	ddsds	dssss	sddxs
2	sssss	sssss	sssss	ssddx	sssss
3	sssds	sssds	ssdss	ddsss	sdsss
4	sssss	sssss	sssss	sssxs	sssss

(b)

Figure2: Example for filling selection register

IV.EXPERIMENTAL RESULTS

Experiments were performed using proposed method with four ISCAS 89 benchmark circuits. The results are shown in Table I. The 0.5% of X's are randomly assigned in the test responses-specifically, 90% of unknowns from 10% of the scan chains. The first column shows the benchmark circuits. The remaining columns give the comparison of the compression ratio for X-canceling, X-masking and X-canceling, and proposed method, respectively. It can easily see that the proposed method provides the significant reduction of test data overhead in all cases.

Table I: Compression ratio with 0.5% unknowns

Circuit	X-Canceling	X-Masking and X-canceling	Proposed method
s15850	19.8x	10.5x	**35.4x**
s13207	24.2x	30.1x	**69.8x**
s38584	23.1x	14.8x	**39.2x**
s38417	23.2x	26.7x	**66.9x**

V.CONCLUSION

Due to the large different distribution of X's in different scan chains in the real designs. By observe the positions of both d's and X's in a set of test patterns, the values of the selection register and mask data can be calculated for dividing X's into two groups; a large number of X's and few X's. The reiterative X-masking logic is inserted between the output of scan chains and the input of X-canceling MISR. The purpose of the reiterative X-masking logic is to handle large number of X's with smallest amount of mask data. The rest of few X's can be handled directly by the X-canceling MISR.

The experimental results were shown that the proposed method provides significantly improvement on the amount of compression. By inspect the distribution of X's, the proposed method will give more effective when the majority of X's is located in small parts of scan chains (no more than 10% of all scan chains).

REFERENCES

[1] L.T. Wang, C.-W. Wu, X. Wen, *VLSI Test Principles and Architectures*, Morgan Kaufmann,2006.

[2] Mitra, S., and K.S. Kim, "*X-Compact: An Efficient Response Compaction Scheme*," IEEETrans. on Computer-Aided Design, Vol. 23, No. 3,pp. 421-432, Mar. 2004.

[3] K. K. Saluja and M. Karpovsky, "*Testing computer hardware through data compression in space and time*," in Proc. ITC, 1983, pp. 83-88.

[4] Rajski, J., J. Tyszer, C. Wang, and S.M. Reddy, "*Finite Memory Test Response Compactors for Embedded Test Applications*," IEEE Trans. On Computer-Aided Design, Vol. 24, No. 4, pp. 622-634,Apr. 2005.

[5] J. Rajski et al, "*Convolutional compaction of test responses*," in *Proc. ITC*, 2003, pp. 745-754.

[6] V. Chickermane, B. Foutz, and B. Keller, *Channel Masking Synthesis for Efficient On-Chip Test Compression*, IEEE International Test Conference,pp.452-460, 2004.

[7] Barnhart, C., V. Brunkhorst, F. Distler, O.Farnsworth, B. Keller, and B. Koenemann, "*OPMISR: the Foundation for Compressed ATPG Vectors*," Proc. of International Test Conference , pp.748-757, 2001.

[8] Pomeranz, I., S. Kundu, and S.M. Reddy, "On Output Response Compression in the Presence of Unknown Output Values," Proc. of Design Automation Conference, pp. 255-258, 2002.

[9] Naruse, M., I. Pomeranz, S.M. Reddy, and S. Kundu, "*On-Chip Compression of Output Responses with Unknown Values Using LFSR Reseeding*," Proc. of International Test Conference, pp. 1060-1068, 2003.

[10] Richard putmen, "*Using reiterative LFSR based X-masking to increase output compression in presence of unknowns*" GLSVLSI'08, , Orlando, Florida, USA May 4–6, 2008

[11] Touba, N.A., "X-Canceling MISR – An X-Tolerant Methodology for Compacting Output Responses with Unknowns Using a MISR," Proc. of International Test Conference, Paper 6.2, 2007.

[12] S. Mitra, and K. S. Kim, *X-Compact: an efficient response compaction technique for test cost reduction*, IEEE International Test Conference, pp.311-320, 2002

[13] P. Wohl, J. A. Waicukauski, S. Patel, and M. B. Amin, *X Tolerant Compression and Application of Scan-ATPG Patternsin a BIST Architecture*, IEEE International Test Conference,pp.727-736, 2003.


ECTI-CON 2011
KHON KAEN UNIVERSITY
8th
Electrical Engineering/ Electronics, Computer, Telecommunications and Information Technology (ECTI) Association, Thailand - Conference 2011
Khon Kaen, Thailand
May 17-19, 2011
Pullman Khon Kaen Raja Orchid Hotel
ECTI Association
KHON KAEN UNIVERSITY
IEEE
THAILAND SESSION

A Selective X-masking for Test Responses in the Presence of Unknown Values

Xu shubing[1], Taweesak Reungpeerakul[2]
Department of Computer Engineering, Faculty of Engineering
Prince of Songkla University, Hatyai ,Songkhla 90112
5110120131@email.psu.ac.th[1], rtaweesak@coe.psu.ac.th[2]

***Abstract*-As the unknown values (X's) appear in the output response, a large amount of scan cells may not be observed during test. In this paper we present a flexible X-masking logic called selective X-masking to handle the majority of X's, while the X-tolerant compactor is applied to handle the remained X's. The objective of this approach is to improve the observability of the test responses. The results show that this scheme can achieve a significant improvement of observability for output responses and reduce significantly the data overhead of the X-masking.**

Key words*-*X-masking, Compactor, selective

I. INTRODUCTION

The scan test has been a greatly applied approach in design-for-test (DFT) so far. The test data stored in ATE (automatic test equipment) for stimulus compression and output response compaction determines the total test data overhead, and the test application time depends on the length of scan chains [1]. In order to reduce test cost in industry design, the initial long scan chains are also cut into large number of shorter scan chains. As the increasing huge test data volume in industry design, researches have recently being working on input stimulus compression and output response compaction in order to reduce the number of channels on the ATE, tester memory and tester time. In this work, we focus on the compaction of the output responses. Basically there are three types of compactors: space compactors [2] [3], time compactors [3] and finite memory compactors [4] [5].

For the output response compaction, the occurrences of X's in the test responses have been the greatest barrier to effect the compaction. One of the major issues for the test compaction is how to handle test responses containing X's. The sources of X's are caused by several conditions such as bus contention, uninitiated memory, un-modeled logic, floating tri-states bus, etc. When X's are introduced to the space compactors, the non-X test responses in the current clock cycle going through the compactor are XORed by X's ,then the non-X values are corrupted and the fault coverage may lost. The time compactor, for example the Multiple Input Signature Registers (MISRs), is intolerance of X's. If there is no X's in test responses, it can compact an infinitely long output sequence into a fixed-length signature [6]. Unfortunately, once X's appear in these responses, the signature is corrupted from which the faulty signature could not be distinguished. Finite memory compactors take advantage of combining time and space dimensions. However, as the increasing number of X's in output responses, the compaction ratio may greatly decrease.

Lots of schemes have been proposed to handle X's in output response. The widely used techniques to handle X's are X-blocking [6], X-masking [8] [9] [10] [11] and X-tolerant [2] [4] [12] [13] [14]. The X-blocking scheme needs extra logic in the circuit-under-test (CUT) [2], which can cause fault coverage lost and additional area overhead. Several X-masking techniques are proposed in previous work in [8] [9] [10]. In one of X-masking techniques, the conventional LFSR X-masking scheme [10] guarantees to mask all X's and keeps specified bits (d's) as well, where each value contained by d's is used to detect one or more faults. However, a large amount of mask data is generated for masking every scan slice. Another X-masking scheme is called reiterative X-masking [11], the volume of mask data decreases greatly due to reusing the mask bit. It is necessary to use interval counter for controlling.

Instead of masking X's, X-tolerant schemes have been introduced in [2] [12] [13] [14]. In [2], XOR gates are used to minimize the impact of non-X value being masked by X's. Since it can guarantee to check erroneous compactor output in the presence of limited number of X's, the errors can be detected by the tester when the errors are propagated to the compactor outputs with X's appearing in current cycle. In the schemes of [14], there are N scan chains and the compacted test responses outputs are M, where N>M. In order to detect faults, the test responses from scan chains containing d's and X's are propagated to different compactor outputs. However, the compaction ratio is extremely degraded as the increasing X's and the corrupted outputs by X's may decrease fault coverage.

In this paper, we propose a selective X-masking logic to mask X's in the test cubes introduced in our previous work [15]. The selective X-masking logic is applied to mask the majority of X's, after that the output responses go through to the X-tolerant compactor[13] [16] which can tolerate the remained X's in the test responses. The main objective of this approach is to improve efficiency by masking the scan chains selectively as well as improve the observability of the test responses.

This paper is organized as follow: The Section II gives an overview of the reiterative X-masking. In the Section III, the architecture of a selective X-masking based on compactor is described. Experimental results are shown in the Section IV

following by the conclusion in the section V.

II. Overview of Reiterative X-Masking

Comparing the conventional LFSR X-masking scheme [10] which masks for every scan slice to ensure all X's masked, the reiterative X-masking [11] can reuse masking data for many scan slices. The primary advantage of the reiterative X-masking logic is to reduce the total overhead of X's masking.

Fig. 1 is a simple example to demonstrate the operation of reiterative X-masking. In Fig.1 (a) the mask bit (M) is given for every five (the size of the interval counter) scan slices. There are totally three cases for assigning the mask bit:

1) M is set to '1' (masked) since X's are present without d's in the scan slices.
2) M bits should be set to '0' (unmasked) if and only if d's are present.
3) If there are neither of X's and d's in these scan slices, then the M is set to '?'(don't care).

Scan chains		M		M		M		M
1	dssss	0	dsdss	0	sssds	0	sddsx	0
2	xsxsx	1	sxsxs	1	sdssx	0	sssss	?
3	sdsss	0	ssdss	0	ddsss	0	sssss	?
4	sssss	?	sdsss	0	sxsss	1	sssss	?

(a)

1	sssss	dsdss	sssds	sddsx
2	sssss	sssss	sdssx	sssss
3	sdsss	ssdss	ddsss	sssss
4	sssss	sdsss	sssss	sssss

(b)

Fig.1 .Reiterative X-masking example

The results are shown in Fig. 1(b) where two X's are left. Normally, if the size of interval counter gets smaller, the remaining of X's usually presents less, but more mask bits are required, otherwise the mask bits will be smaller. Thus, the minimum size of interval counter has been determined and can be re-loaded in different test patterns.

III. Architecture of Selective X-Masking for Compactor

In actual designs the distribution of X's has the tendency to be clustered in output responses, which means that the majority of X's may just locate in a small number of scan chains [5]. It is possible to mask as many X's as possible with the smallest mask data overhead. The proposed method arranges all scan chains into two groups. One group contains large amount of X's, which can be handled by the X-mask logic. Another group which consists of the small number of X's can directly go through input of compactor without masking. The main purposes of this scheme are not only to get rid off X's, but also to decrease the mask data overhead as much as possible.

Fig.2 shows the architecture of proposed approach. The architecture of m scan chains contains m-bit mask register, an interval counter, m 2-to-1 Multiplexers (MUX) and m-bit selection register. The selection register is used to determine whether one scan chain can directly be sent to the compactor. For example, if the port '0'of the top MUX is selected by selection register, the top scan chain can directly be sent to compactor. Otherwise, if this value is '1', the scan chain needs to be sent to the mask logic first. As the selection register is only loaded one time during whole test, the overhead of this controlling data is negligible.

Fig.3 demonstrates a simple example on how to assign the values of the selection register. In this figure the mask bit (M) is given for every five scan slices. Since there are few numbers of X's in scan chain 1, 3, and 4, then these scan chains can be connected directly to compactor. However, the scan chain 2 contains six X's, the density of X's in chain 2 is much larger than in the chain 1, 3, and 4, thus the scan chain 2 is connected to the X-masking logic. As a result, the data bit 1, 3 and 4 of selection register equal to '0' (unselected) and bit 2 is set to '1'(selected). Since the scan chain 2 is selected, the mask data of mask register is shown in Fig. 3a.

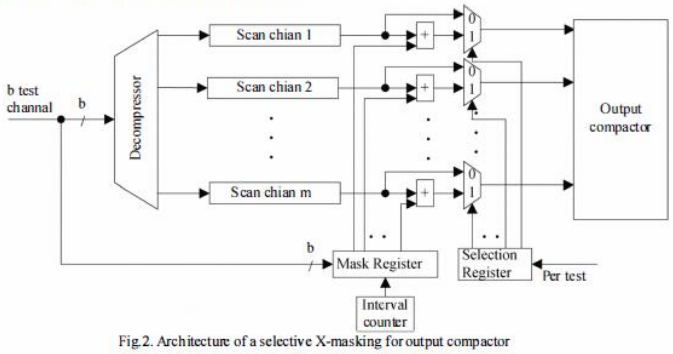


Fig.2. Architecture of a selective X-masking for output compactor

For an interval of five scan slices, mask bit (M) is set to '1' (masked) in scan chain 2 since X's are presented with no d's, mask bit is '0'(unmasked) due to the existence of d's, and '?'(don't care) means neither of X's and d's in scan slices. Since scan chains 1, 3, 4 are not selected, the scan slices in these chains are not concerned to mask. The result after passing through the selective X-masking logic is shown in Fig.3b. The remained three X's can be directly tolerated by an X-tolerant compactor.

Scan chains	M	M	M	M	Selection register
1	d s s s s ?	d s d s s ?	s s s d s ?	s d d s x ?	0
2	x s x s x 1	s x s x s 1	s d s s x 0	s s s s s ?	1
3	s d s s s ?	s s d s s ?	d d s s s ?	s s s s s ?	0
4	s s s s s ?	s d s s s ?	s x s s s ?	s s s s s ?	0

(a)

1	d s s s s	d s d s s	s s s d s	s d d s x
2	s s s s s	s s s s s	s d s s x	s s s s s
3	s d s s s	s s d s s	d d s s s	s s s s s
4	s s s s s	s d s s s	s x s s s	s s s s s

(b)

Fig.3.Selective X-masking example

IV. Experimental Results

Experiments are performed on four ISCAS89 benchmark circuits to implement the proposed method. In Table 1, the first column lists the circuit used in this experiment. The second column gives the total number of the scan elements. The third column lists the number of scan chains. The last column shows the number of test patterns.

Table1. ISCAS 89 benchmark circuit

Circuit	Scan element	# of chain	# of test pattern
s15850	534	107	229
s13207	638	126	343
s38417	1636	164	485
s38584	1426	143	464

We report the results to show how the selective X-masking logic can improve the percentage of observability for output responses (%of O. response) for the X-compactor (one of X-tolerant compactors). In this experiment, 1% of X's are randomly generated in the test responses-specifically, 90% of X's filled in 10% of the scan chains.

In Table 2, in the second column it shows the percentage of X's masked in the test cubes by the selective X-masking logic (S. mask). The total overhead of mask bits is shown in the third column. The fourth column (fifth column) shows the percentage of observability for output responses with (without) selective X-masking logic. The sixth column is the improvement of observability for output responses. By comparing the fourth and fifth columns, the percentage of observability for output responses is improved in the range of 7.93% to 14.32% (11.67% on the average for four circuits).The results indicate that the selective X-masking logic can achieve a significant improvement of observability for output responses in the comparison.

Circuit	X-compactor				
	With S. mask			W/o S. mask	Improve-ment% (a-b)/a
	%X's mask	Total Mask bits	% of O. Response(a)	% of O. Response(b)	
s15850	64.8	1,511	**96.33**	82.53	14.32
s13207	80.9	2,130	**98.85**	85.29	13.73
s38417	83.6	7,658	**99.63**	88.98	10.69
s38584	65.2	8,024	**97.71**	89.96	7.93
Avg.	73.6		**98.13**	84.60	11.67

Table 2.Comparison between with and without using selective X-masking with 1% X's

When the number of X's has been increased to 2%, the percentage of observability was enhanced by 34.11% on average for four circuits as shown in Table 3.

Circuit	X-compactor				
	With S. mask			W/o S. mask	Improve-ment% (a-b)/a
	%X's mask	Total Mask bits	% of O. Response(a)	% of O. Response(b)	
s15850	63.1	1,854	**89.05**	56.22	36.87
s13207	81.6	3,187	**96.80**	61.55	36.42
s38417	83.9	11,171	**98.58**	69.34	29.66
s38584	64.8	10,316	**92.20**	61.34	33.47
Avg.	73.4		**94.16**	62.11	34.11

Table 3.Comparison between with and without using selective X-masking with 2% X's

In Tables 4 and 5, we show the results comparing the overhead of mask bits between our approach and reiterative X-masking [11] in the presence of 1% and 2% of X's. As the results show, the reduction of mask bits is 77.67% and 72.20% on average, respectively.

Circuit	Mask bits		Reduction %(b-a)/b
	Our approach (a)	Reiterative X-masking [11] (b)	
s15850	1,511	9,532	84. 14
s13207	2,310	7,937	73.16
s38417	7,658	29,732	74.32
s38584	8,024	38,303	79.05

Table 4.The overhead of mask bits based on 1% of X's

Circuit	Mask bits		Reduction %(b-a)/b
	Our approach (a)	Reiterative X-masking [11] (b)	
s15850	1,854	9,975	81.41
s13207	3,184	9,199	65.38
s38417	11,171	33,964	67.10
s38584	10,316	41,122	74.91

Table 5.The overhead of mask bits based on 2% of X's

V. Conclusion

In this paper the selective X-masking is proposed due to the large different distribution of X's in scan chains in actual

designs. In order to reduce overhead of mask bits, the selection register separates all scan chains into two groups (large density of X's group and low density of X's group), and the masking logic just handles large density of X's group. After that the remaining number of X's can be tolerated by an X-tolerant compactor. The experimental results were shown that the proposed method can achieve significant improvement of observability for scan cells and reduce the X-masking data overhead significantly. By inspect the distribution of X's, the proposed method will give more effective when the majority of X's is located in small parts of scan chains.

REFERENCE

[1] M.C.-T.Chao,S.wang, and S.T.Chakredhar, K.T.Cheng "Response Shaper:A Novel Technique To Enhance Unknown Tolerance for Output Response compaction," *IEEETrans. on Computer-Aided Design, ,pp. 80-87, Nov. 2005.*

[2] Mitra, S., and K.S. Kim, "X-Compact: An Efficient Response Compaction Scheme," *IEEETrans. on Computer-Aided Design, Vol. 23, No. 3,pp. 421-432, Mar. 2004.*

[3] K. K. Saluja and M. Karpovsky, "Testing computer hardware through data compression in space and time," *in Proc. ITC, pp. 83-88, 1983.*

[4] Rajski, J., J. Tyszer, C. Wang, and S.M. Reddy, "Finite Memory Test Response Compactors for Embedded Test Applications," *IEEE Trans. On Computer-Aided Design, Vol. 24, No. 4, pp. 622-634, Apr. 2005.*

[5] J. Rajski et al, "Convolutional compaction of test responses," *in Proc. ITC, pp. 745-754, 2003.*

[6] L.T. Wang, C.-W. Wu, X. Wen, VLSI Test Principles and Architectures, *Morgan Kaufmann, 2006.*

[7] V. Chickermane, B. Foutz, and B. Keller, Channel Masking Synthesis for Efficient On-Chip Test Compression, *IEEE International Test Conference,pp.452-460, 2004.*

[8] Barnhart, C., V. Brunkhorst, F. Distler, O.Farnsworth, B. Keller, and B. Koenemann, "OPMISR: the Foundation for Compressed ATPG Vectors," *Proc. of International Test Conference , pp.748-757, 2001.*

[9] Pomeranz, I., S. Kundu, and S.M. Reddy, "On Output Response Compression in the Presence of Unknown Output Values," *Proc. of Design Automation Conference, pp. 255-258, 2002.*

[10] Naruse, M., I. Pomeranz, S.M. Reddy, and S. Kundu, "On-Chip Compression of Output Responses with Unknown Values Using LFSR Reseeding," *Proc. of International Test Conference, pp. 1060-1068, 2003.*

[11] Richard putman, "Using reiterative LFSR based X-masking to increase output compression in presence of unknowns" *GLSVLSI'08, , Orlando, Florida, USA May 4–6, 2008.*

[12] Touba, N.A., "X-Canceling MISR – An X-Tolerant Methodology for Compacting Output Responses with Unknowns Using a MISR," *Proc. of International Test Conference, Paper 6.2, 2007.*

[13] S. Mitra, and K. S. Kim, "X-Compact: an efficient response compaction technique for test cost reduction", *IEEE International Test Conference, pp.311-320, 2002.*

[14] P. Wohl, J. A. Waicukauski, S. Patel, and M. B. Amin, "X Tolerant Compression and Application of Scan-ATPG Patternsin a BIST Architecture", *IEEE International Test Conference,pp.727-736, 2003.*

[15] X.shubing and T. Reungpeeraku, "Hybrid Selective X-masking and X-canceling MISR for Test Responses", *ITC-CSCC, pp.347-350, 2010.*

[16] T.Rabenalt and M.Goessel and A. Leininger "Masking of X-values by Use of a Hierarchically Configurable Register", *European Test Symposium, pp.149-154, 2010.*

VITAE

Name Mr. Xu Shubing

Student ID 5110120131

Educational Attainment

Degree	Name of Institution	Year of Graduation
Bachelor of Engineering	JiangXi University of Science and Technology	2008

List of Publication and Proceedings

[1] X. shubing and T. Reungpeerakul, "A selective X-masking for test responses in the presence of unknown values" *Proceedings of ECTI-CON 2011*, pp.159-162, KhonKaen, Thailand, May, 2011.

[2] X. shubing and T. Reungpeerakul, "Hybrid Selective X-masking and X-canceling MISR for Test Responses", *Proceedings of ITC-CSCC 2010*, pp.347-350, Thailand July, 2010.