Summer 8-4-2021

REDUCING POWER DURING MANUFACTURING TEST USING DIFFERENT ARCHITECTURES

Yi Sun
yis@smu.edu

REDUCING POWER DURING MANUFACTURING TEST USING DIFFERENT ARCHITECTURES

Approved by:

Jennifer Dworak
Electrical & Computer Engineering
Dissertation Committee Chairperson

Sukumaran Nair
Electrical & Computer Engineering
Southern Methodist University

Theodore Manikas
Computer Science
Southern Methodist University

Ping Gui
Electrical & Computer Engineering
Southern Methodist University

Gary Evans
Electrical & Computer Engineering
Southern Methodist University

Kundan Nepal
Electrical & Computer Engineering
University of St. Thomas

REDUCING POWER DURING MANUFACTURING TEST USING DIFFERENT ARCHITECTURES

A Dissertation Presented to the Graduate Faculty of the

Lyle School of Engineering

Southern Methodist University

in

Partial Fulfillment of the Requirements

for the degree of

Doctor of Philosophy

with a

Major in Computer Engineering

by

Yi Sun

M.S.,EE, Lyle school of Engineering
B.S.,EE , Chongqing University

August 4, 2021

ACKNOWLEDGMENTS

This work is supported by NSF CCF1812777 and CCF1814928. I am deeply grateful for their financial support.

I thank my advisor, Jennifer Dworak, for her continuous support and patience whenever I am stuck in a problem. Whenever I thought I am not designed for being a PhD, it is her support and direct guidance help me hone my problem-solving skills and greatly improved my writing skills. This forever change me into a critical thinker instead of being a technician.

I would also like to show my gratitude to (Kundan Nepal, Department Chair, University of St. Thomas) and (Theadore Manikas, professor, Southern Methodist University) for sharing their pearls of wisdom with me during the course of my research, especially regarding tool usage and manuscripts. I also want to thank the reviewers of the conferences and Journal for their insights in my paper. I also really appreciate all the suggestions from the committee members to help me perfect my dissertation.

Last but not least, I thank my family and friends for their love and support during the course of my study in pursuit of a PhD.

Sun, Yi

M.S.,EE, Lyle school of Engineering, 2013
B.S.,EE, Chongqing University, 2009

Reducing power during manufacturing test using different architectures

Advisor: Jennifer Dworak

Doctor of Philosophy conferred August 4, 2021

Dissertation completed August, 2021

Power during manufacturing test can be several times higher than power consumption in functional mode. When operating in test mode, the circuit's internal scan chains are generally either operating in shift mode or capture mode. When in shift mode, the circuit's state can be set to an arbitrary value by shifting the desired values into the flops. When operating in capture mode, the circuit's flops are set by the normal functionality of the circuit. As a result, the power during test can also be divided into two categories: shift power and capture power. In this dissertation, I have described three different architectures to reduce shift and/or capture power during test as well as in certain scenarios, including field test.

In the first architecture, scan chains are divided into several segments. The segments can be enabled or disabled during capture mode to save power. Every segment needs a control bit to enable capture in a segment when new faults are detectable on that segment for that pattern. Otherwise, the segment should be disabled to reduce capture power. We group the control bits together into one or more control chains. We will show this approach can dramatically reduce power in many circuits.

Two significant disadvantages of the first architecture include 1) the likely need for one or more extra pins to shift data into the control chain, and 2) the need for significant post-processing of fault simulation data to determine the appropriate values of the control bits on each pattern. To address these issues, we explored a second architecture, that stitches the control bits into the chains they control as EECBs (embedded enable capture bits) in

between the segments. This allows an ATPG software tool to automatically generate the appropriate EECB values for each pattern to maintain the fault coverage. This works even in the presence of an on-chip decompressor.

The last architecture presented in this dissertation focuses primarily on the self-test of a device in a 3D stacked IC when an existing FPGA in the stack can be programmed as a tester. We explore the total energy expenditure during scan shift that is needed when very short chains are fed by high bandwidth TSV connections from the FPGA to the circuit under test. We show that the energy expended is significantly less than would be required using low power patterns fed by an on-chip decompressor for the same very short scan chains.

All of the proposed approaches can be utilized at manufacturing and while testing circuits in the field to significantly reduce the total energy used during test.

TABLE OF CONTENTS

LIST OF FIGURES x

LIST OF TABLES xiii

CHAPTER

1. Introduction 1

2. Background of Manufacturing Test and FPGA 7

2.1. Manufacturing Test 7

2.1.1. Scan Design 9

2.1.1.1. Scannable D-flip-flop 9

2.1.1.2. Scan Chain 11

2.1.1.3. Test Procedure 11

2.1.2. Built in Self Test 13

2.1.2.1. The Utilization of an FPGA for Test 14

2.1.3. Test Data Compression 15

2.2. Fault Models 16

2.2.1. Stuck-at Fault Model 17

2.2.2. Transition Fault Model 18

2.2.2.1. Launch-off-Shift Method (LOS) 19

2.2.2.2. Launch-off-Capture Method (LOC) 19

2.3. Test Power Issues 20

3. Reducing Capture Power through the Use of a Control Chain 23

3.1. DFT Architecture for Segment-based Capture Control 23

3.2. Methodology 25

3.3. Results 28

3.4. Capture Power Reduction Based on Forward Simulation 29
3.5. Additional Capture Power Reduction based on Reverse Simulation 33
3.6. Area overhead and delay 34
3.7. Conclusion 35
4. Low Power Test through ATPG-Configured Embedded Enable Capture Bits . . 38
4.1. Second DFT Architecture for Segment-Based Capture Control 38
4.2. EECB Insertion Procedure 43
4.3. Results 45
4.3.1. Stuck-At Fault Model Results 46
4.3.1.1. Unconstrained patterns generated with the ATPG tool . . 46
4.3.1.2. All segments are forced to capture for the first few patterns 49
4.3.2. Transition Fault Model Results 51
4.3.2.1. Unconstrained patterns generated with the ATPG tool . . 51
4.3.2.2. All segments are forced to capture for the first few patterns 53
4.3.3. Overhead 53
4.3.3.1. Area Overhead 53
4.3.3.2. Test Time Overhead 55
4.4. Conclusion 55
5. FPGA based Tester to Enhance Field-Testing in a 3D Stacked IC 57
5.1. Using an FPGA for low power testing of other dies in a 3D stack 57
5.2. Exploiting FPGA's Generic Architecture 60
5.3. ADJCOM 62
5.3.1. Adjacent fill merging algorithm 62
5.3.2. FPGA Implementation Results and Analysis 66
5.3.3. Data Reduction using ADJCOM 70

5.3.4. Switching Activities 73
5.4. XRET 74
5.4.1. Merging with "X" in pattern set retained 74
5.4.2. Data Reduction using XRET 77
5.4.3. Switching activities 78
5.5. Test Response and Test Time 80
5.5.1. Conclusions 82
6. Conclusions 83
BIBLIOGRAPHY 85

LIST OF FIGURES

Figure Page

1.1. 3D stacked IC 6

2.1. Manufacturing test of a circuit. [1] 8

2.2. Original circuit with no scan design. 9

2.3. Scan design. 10

2.4. Scannable D-flip-flop. 10

2.5. Scan chain 11

2.6. Scan Testing Procedure. 12

2.7. Overview of the BIST architecture [2] 13

2.8. 3D stacked IC, which has shown before in the previous chapter 14

2.9. The overall architecture of Test Data Compression. [2] 16

2.10. On Chip Decompressor [3] 17

2.11. Waveform for Launch-off-Shift delay test [2]. 19

2.12. Waveform for Launch-off-Capture delay test [2]. 20

3.1. Schematic design of segmented scan chain. 24

3.2. Procedure to determine which segment to keep when faults are detected in multiple segments. 27

3.3. Average % reduction in toggle count. 30

3.4. Percentage of segments disabled during capture for *fpu_double* circuit with overhead of 3.4% 31

3.5. Toggle reduction per pattern during capture for *fpu_double* circuit with overhead of 3.4% 32

3.6. Correlation between toggle reduction and segments disabled. 32

3.7. Reverse Simulation Flow 36

3.8. Comparison of toggle reduction between original flow and Backwards Simulation flow. 37

4.1. Schematic design of regular scan chain with enable signal. 38

4.2. Schematic design of segmented scan chain. 39

4.3. Shift in random patterns into a regular scan chain. 41

4.4. Shift in low power patterns into a regular scan chain 41

4.5. Shift in low power patterns into the segmented scan chain 41

4.6. Flow chart of EECB insertion procedure. 44

4.7. *des56* Test Power Reduction for Stuck-at Fault Patterns 46

4.8. *fm_receiver* Test Power Reduction for Stuck-at Fault Patterns 47

4.9. *colorconv* Test Power Reduction for Stuck-at Fault Patterns 48

4.10. *fpu_double* Test Power Reduction for Stuck-at Fault Patterns 48

4.11. *des56* Test Power Reduction for Stuck-at Fault Patterns when all the segments are enabled during capture for the first n patterns. Each colored bar corresponds to a different value of n. (Segment length = 32 bits.) 49

4.12. Total number of patterns and corresponding switching activity when all segments are forced to capture for the first few patterns in the pattern set for *des56* 50

4.13. *des56* Test Power Reduction for Transition Fault Patterns 51

4.14. *colorconv* Test Power Reduction for Transition Fault Patterns 52

4.15. *fpu_double* Test Power Reduction for Transition Fault Patterns 52

4.16. *fm_receiver* Test Power Reduction for Transition Fault Patterns 52

4.17. *fm_receiver* Total test power for transition fault patterns with regard to pattern count for different amounts of scan chain segmentation 53

4.18. % Test time overhead for stuck-at and transition fault patterns for *fpu_ double* . 56

5.1. Example FPGA-based implementation for storing pattern data for a single scan chain. This is repeated for multiple chains, with LUTs possibly shared among chains. 61

5.2. Flowchart for LUT and Select Line Reduction using ADJCOM. 64

5.3. Resulting implementation for patterns shown in Table II after pattern merging. 66

5.4. FPGA-based tester block diagram. 68

5.5. Switching activities for our method vs. on chip decompressor. 73

5.6. Flowchart for LUT and Select Line Reduction using XRET. 75

5.7. Resulting implementation for patterns shown in Table I after pattern merging. 76

5.8. Switching Activities for our methods vs. on-chip decompressor 79

5.9. Switching activities in IC floorplan for *fpu* using XRET versus an on-chip decompressor. The legend shows switching activity scale – red corresponds to high switching and purple/blue low switching activity. 80

5.10. Test time reduction:our Method vs. on chip Decompressor. 82

LIST OF TABLES

Table Page

3.1. Benchmark circuit properties 26

3.2. Toggle count distribution across the 5 chains in *fpu_double*. 33

3.3. Area Overhead and Timing Delay with extra control chain added 34

4.1. Number of scan cells per segment per chain 44

4.2. Characteristics of Benchmark Circuits 45

4.3. % Area overhead with different segment lengths 54

5.1. Example original pattern set 63

5.2. Example Pattern Data after Adjacent Fill 63

5.3. Characteristics of our Opencores.org Benchmark Circuits 67

5.4. Experiment 1: All modules are distributed RAMs/Slice LUTs 69

5.5. Experiment 2: Mux select lines implemented in block RAMs (BRAMs) 69

5.6. Data Storage Reduction 71

5.7. Experiment 1—All modules are distributed RAMs/Slice LUTs 77

5.8. Experiment 2— Mux select lines implemented in block RAMs (BRAMs) . . . 78

5.9. Data Storage Reduction 78

5.10. Test coverage before and after using MISR 81

Chapter 1
Introduction

The manufacturing process of a chip is very complex. The first step is to harvest the silicon from nature and form it into a cylinder ingot and cut the ingot into multiple thin layered pieces called wafers. The next step is to allow the photolithographic "printing" process to form a die's multilayered transistors and interconnects corresponding to electrical circuits on every wafer. Hundreds of devices (e.g. processors) can be created together on a single silicon wafer. Finally, the wafer can be cut into pieces, where each piece corresponds to a single die ready for packaging. The outer package of a die protects the die and delivers critical power and electrical connections. The packaged die is now called a chip. It can be placed directly onto a circuit board and can then be used multiple contexts (e.g. in mobile devices such as the circuit board in the smartphone or tablet).

Unfortunately, the fabrication process is imperfect. Manufacturing test is thus performed to verify whether the design was manufactured correctly (i.e. whether the chip is a working part). In order to verify it is a working chip, a software tool called ATPG (automatic test pattern generation) is used to generate test patterns. A test pattern corresponds to the values (e.g. logic 0's or logic 1's for digital circuits) applied to the circuit under test, which enables the test engineer to distinguish between a circuit that is operating correctly and a circuit that displays defective circuit behavior. Test equipment, such as automatic test equipment (ATE), is utilized to apply test patterns and check the circuit's responses automatically.

While finding appropriate test patterns for combinational circuits is relatively straightforward, developing a test set for sequential circuits is more difficult. In particular, the response of a sequential circuit depends on both the input pin values as well as the values in the sequential storage elements, such as the flip-flops, in the circuit. To test for particular potential defects, appropriate values need to reside in these sequential elements (flip-flops). If these values are to be placed in those flip-flops using the normal operation of the circuit,

a large amount of backtracking over many clock cycles often must be performed by the sequential ATPG tool that is generating the test patterns. This process takes quite a long time and is hard to perform successfully.

To make testing a chip easier, DFT (design for test) techniques are widely used to improve the testability of a design. One of the most commonly used DFT techniques is called scan design. To make a sequential circuit into a full-scan circuit, all the sequential elements are serially connected to each other to form one or more scan chains. During test, the sequential elements in the scan chain can be set to 0 or 1 by shifting the appropriate values into the sequential elements one at a time on every clock cycle. Depending on the number of flip-flops in the circuit, multiple scan chains can be inserted instead of a single chain to reduce the chain length and thus the number of clock cycles the circuit needs for scan shift. (All the scan chains can potentially be shifted at the same time.)

Thus, the following steps are used to test the full-scan circuit. First, the ATE chooses the next test pattern in the pattern set to apply. The circuit is placed in *shift mode* by asserting the "scan enable" signal, and values corresponding to the selected test pattern are shifted into the scan chain to set up the test. The ATE also applies the appropriate values directly to the input pins. The combinational logic in the circuit will then operate as a function of the values in the flip-flops and at the inputs. New logic values calculated by the circuit will appear at the functional inputs to the flip-flops and at the circuit outputs. The scan enable signal is de-asserted, and the values at the inputs to the flip-flops are captured in the flip flops. This is called *capture mode.* (Multiple capture cycles may be used for some tests.) The output values are sampled, and the circuit is placed back in shift mode to shift out the values captured by the flip-flops from the circuit logic. These values can then be compared to the expected values to determine whether that copy of the circuit has passed that test pattern. While the captured values are shifted out, the next pattern can be shifted in to set up the next test pattern to be applied. This process repeats until all of the test patterns have been applied (or at least until the circuit fails a test and is shown to be defective.)

Unfortunately, over time, test has become more difficult and costly. In particular, while circuit scaling has led to significant increases in the number of transistors integrated per square inch on a chip and thus allows chips with higher performance to be manufactured,

more test patterns are needed to test them—increasing test time. In addition, with more test patterns, more energy is also consumed during test. Furthermore, when tests attempt to provide as much defect coverage as possible with as few patterns as possible, the amount of switching activity and power drawn is often much higher during test than during normal functional operation.

This power and energy consumption during test is a significant problem. For example, excessive power draw during test can cause "IR-drop," which tends to increase circuit delays. As a result, some good dies may fail a test that expends too much power even though they would operate correctly under normal conditions—causing yield loss as good chips are discarded. Excessive power draw can also cause integrated circuits (ICs) to have hot spots, and in some cases even damage or reduce the life expectancy of a chip.

Historically, a significant portion of the low power test research has focused on power draw during scan shift because a significant fraction of the test time and test power is expended during shift. There are many proposed approaches to reduce scan shift power, such as adjacent fill (e.g. [4]), which is one of the easiest to implement. Adjacent fill takes advantage of the fact that, when generating test patterns, there are many don't care bits that can be set to either 0 or 1 in the scan chains while still allowing the targeted fault to be detected. The adjacent fill approach will fill the don't care bits with the same value as the bit next to them. This approach is advantageous in that it allows certain defects to be targeted deterministically with the chosen ATPG algorithm in the normal way, and then fills the remaining don't care bits with values that reduce power draw during shift.

In contrast, reducing capture power is less straightforward. Even though only a few flip-flops in the scan chain may be used to detect a new, as-of-yet undetected targeted fault, many additional flip-flop values may change value during capture. Changes in the values of those flip-flops then propagate further into the circuit during capture mode. The resulting switching activities can create hot spots within a chip.

The test power problem becomes even harder in the presence of an on-chip decompressor. When such a decompressor is used, many of the don't care values are used to accomplish test data compression instead of reducing power draw. ATPG tools still allow patterns to be generated in a low power mode, but the overall reduction that can be achieved must also

satisfy the needs of the decompressor.

As a result, a test architecture in chapter 3 is discussed where a separate control chain has been used to reduce capture power while also being compatible with the on-chip decompressor structure. In this architecture, we focus on significantly reducing capture power without requiring changes to the initial test set, any increase in test time, or any reduction in the accuracy of determining whether a chip is defective or not. We ensure that the DFT hardware is independent of the test generation procedure/test pattern set and thus can be optimized separately.

To implement this architecture, we break a long scan chain into smaller scan segments whose ability to capture data during capture mode is dependent on a bit in a control register. Simple analysis has been done to allow those control bits to be set to enable capture only when the corresponding segment is needed for additional targeted fault detections. With very low flip-flop overhead, we can achieve high power reduction across multiple circuits—even when we start with low-power patterns generated by a commercial tool. For some patterns in the circuits, the power reduction approaches 100%.

Because the investigated architecture requires one or more separate control chains to be added to the existing DFT circuitry, it requires at least one extra pin apart from the on-chip decompressor logic. The extra control pin becomes a problem because the ever smaller sizes of chips will give less and less room for the pins that are needed for the design. (Alternatively, if the size of the chip remains the same, but if the amount of logic on the chip increases with increasing transistor density, pins are still a precious resource because the same number of pins on a package must now access the additional logic.) As a result, we have come up with another architecture to embed the control bits in between the scan segments instead of in a separate chain as was done in the previous architecture. In other words, the extra control bits that were originally in the additional control chain now become part of the existing scan circuitry, which eliminates the use of the extra control pin.

This alternative approach described in chapter 4 also greatly saves on the time spent on post processing as compared to the architecture in chapter 3. Specifically, because of the way the control is implemented, a standard ATPG tool is capable of automatically determining the appropriate values of those control bits to achieve the desired fault coverage. The values

for the control bits can be set by shifting data into the normal functional scan chains in which the control bits are embedded. In addition, this is true even in the presence of an on-chip decompressor. Finally, when inserted into a full-scan design that already contains flip-flops with an enable input (such as for clock gating or sleep modes), no additional logic must be inserted in the paths of the functional logic—minimizing the impact on the functional circuit delay.

While the two previous approaches may be applied to any circuit, reducing test power becomes especially important when testing assembled 3D stacked ICs. Because bare dies are stacked directly on top of each other within the package in a 3D stacked IC, any heat generated by the test is less likely to escape. Thus, even more serious power problems occur. Furthermore, excessive power draw and the corresponding rise in temperature is especially important in field test, where parts may be exposed to environments that have higher temperature than normal due to weather or other heat-producing components in other parts of the system. In field test, having a quick test result in addition to low power test is crucial. As a result, we also explore an architecture to address both issues that can aid in the field test of a 3D stacked IC when an FPGA inserted in the stack is repurposed to apply patterns to other dies in the stack during test.

Using an FPGA as a tester will help in both test time and test power. As shown in Figure 2.8, a 3D stacked IC has many TSVs (through silicon vias) which can be utilized to transmit test patterns into the scan chains on the die under test from the FPGA. Also, because there are many more TSVs in a 3D stacked IC than the number of pins on a board, much shorter scan chains can be inserted to save shift time when the chains can be fed from the FPGA. This is an improvement over using an on-chip decompressor when the scan chains are very short because very short chains require more patterns to be generated and applied when an on-chip decompressor is used [3]. Increasing the number of patterns would increase the total energy expended during test and lead to an increase in temperature—especially at the location of the decompressor itself. The fact that industry has already incorporated FPGAs into some 3D stacks, makes the explored architecture especially reasonable in those stacks because additional dies do not need to be added for test. To program the FPGA as an efficient tester, we devised an intelligent way to store and apply the test patterns

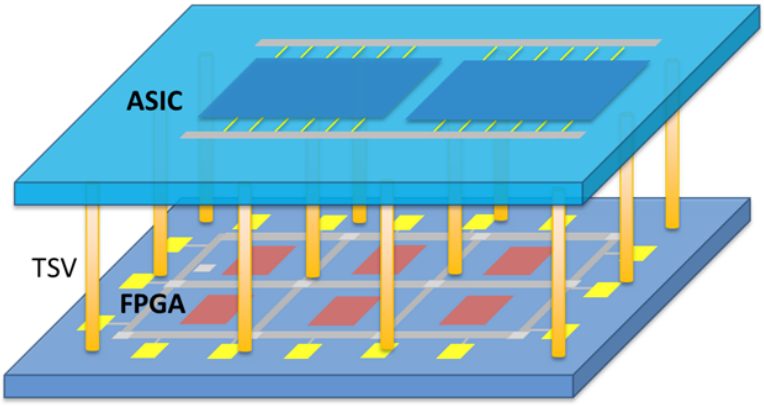

Figure 1.1: 3D stacked IC

in the underlying structure of the FPGA—allowing us to bypass the widely used on-chip decompressor structure. In this work we focus primarily on reducing shift power.

Thus, this dissertation presents three architectures for reducing power during test in VLSI circuits by reducing switching activities during shift cycles or capture cycles or both. The rest of the dissertation is organized as follows. Chapter 2 This chapter introduces some important concepts in manufacturing test, FPGA as well as all the background that the reader needs for a better understanding of the later chapters in this dissertation. Chapter 3 introduces an architecture to reduce capture power in an on chip decompressor test environment [3] with low overhead in test area and volume of test data, and a significant reduction in capture power consumption. Chapter 4 introduces a new architecture where the extra control bits are embedded in between the scannable D-flip-flops (scan DFFs) with little to no effort required for post-processing the patterns from the ATPG tool. Chapter 5 introduces a FPGA based tester in 3D stacked ICs and describes its advantages in reducing power compared to the widely used test data compression method on chip decompressor in a 3D stacked IC. This is approach is especially appropriate in field test. Chapter 6 outlines a conclusion of the methods investigated in this dissertation.

Chapter 2

Background of Manufacturing Test and FPGA

This chapter introduces some important concepts in manufacturing test, FPGA as well as all the background that the reader needs for a better understanding of the later chapters in this dissertation.

2.1 Manufacturing Test

Manufacturing test is the process to verify whether a chip is manufactured correctly. It is performed after a circuit comes out of the manufacturing line and is used for filtering defective parts. To make manufacturing test more effective and efficient, test patterns (sometimes called test vectors) for a digital circuit are generated by an ATPG tool by targeting *faults*. A fault is a "model" of a defect that represents the defect's behavior and effect on the logical operation of a circuit. If a circuit has no faults, we call it a fault-free circuit. When a certain set of values (e.g. test pattern) is applied to the inputs of the fault-free circuit, the output values from the circuit are called the expected response.

Figure 2.1 shows the basic flow of manufacturing test with its three basic components as follows:

- *Circuit under test (CUT)*: This is the circuit being tested.
- *The test application circuitry of the automatic test equipment (ATE)*: It is used to apply test patterns and collect responses from the output pins of the CUT so that they can be compared to the expected test response.
- *The memory of the automatic test equipment (ATE)*: This is used to store the test patterns and the expected test response.

As shown in Figure 2.1, the ATE will apply test patterns to the circuit. Then the CUT's response is analyzed. If the CUTs' response are the same as the response of a fault free

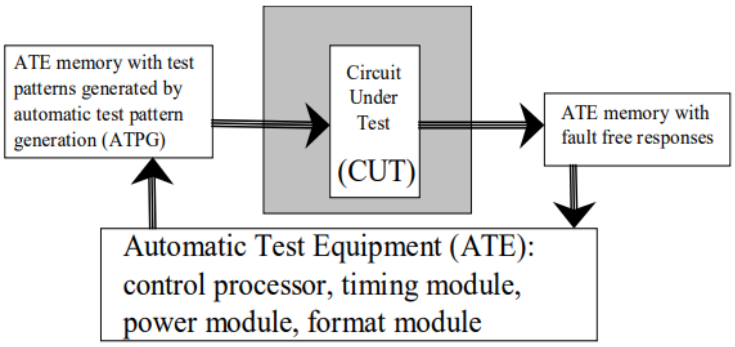

Figure 2.1: Manufacturing test of a circuit. [1]

circuit. Then the circuit is considered working correctly for that particular test pattern. The responses are also stored in the ATE for other purposes such as diagnosis.

Testing of a combinational only circuit is relatively easy. Once the value of the primary inputs has been set, the corresponding primary outputs are also determined and can be observed accordingly.However, if testing a sequential circuit is considered, which has sequential elements such as flip-flops, it can become very complicated. Figure 2.2 shows the Huffman model [5] that represent the structural of a sequential circuit. Sequential elements (DFFs in Figure 2.2) in the circuit need to be set to the desired values because the output of a sequential circuit does not only depends on the values of the input pins, but also on the current state of the circuit (the values in the sequential elements). In this case, the ATPG (automatic test pattern generation) tool needs to create test patterns over many clock cycles to make sure the sequential circuit is in the state the tool expects it to be. As a result, sequential ATPG is much harder to implement compared to combinational ATPG. Both the run-time and complexity to generate the test patterns can increase drastically. Thus, it has also been found to be impractical for large circuits [2].

To make it easier to create and apply effective tests, DFT (Design for Test) circuitry has been developed. DFT techniques improve testability by increasing the controllability and observability of the circuit logic. The most popular DFT techniques for testing VLSI circuits include scan design, Built-In Self-Test (BIST), and test data compression. We briefly describe all of these DFT techniques in the following subsections because they are necessary

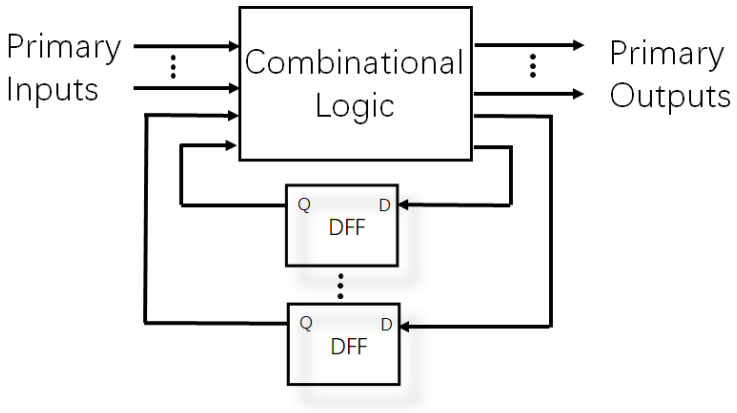

Figure 2.2: Original circuit with no scan design.

to understand the investigated approaches described in the following chapters.

2.1.1 Scan Design

The purpose of the scan design is to more easily access the sequential logic elements in the circuit under test. This is achieved by replacing the D-flip-flops (DFFs) in Figure 2.2 with a scannable D flip-flop (SDFF) design, such as that shown in Figure 2.4. The overall circuitry with the scan chain inserted then becomes Figure 2.3.

2.1.1.1 Scannable D-flip-flop

To easily apply predetermined values to the sequential elements (flip-flops) during the scan-based testing procedure, every flip-flop in the circuit will have a mux added before the **D** input. It thus becomes a scannable D-flip-flop known as a Mux-D Scannable D-flip-flop. Figure 2.4 shows such a flip-flop. When the scan enable (**SE**) signal is 0, and the clock signal rises, the value at D_n is "captured" in the flip-flop, and the captured value appears on the **Q** output. The value will remain the same at **Q** at least until the next rising clock edge. When **SE** is 1, the value at the scan data input (**SDI**) is captured by the flip-flop on the next rising clock edge. **SE** is set to 1 when the circuit is in shift mode.

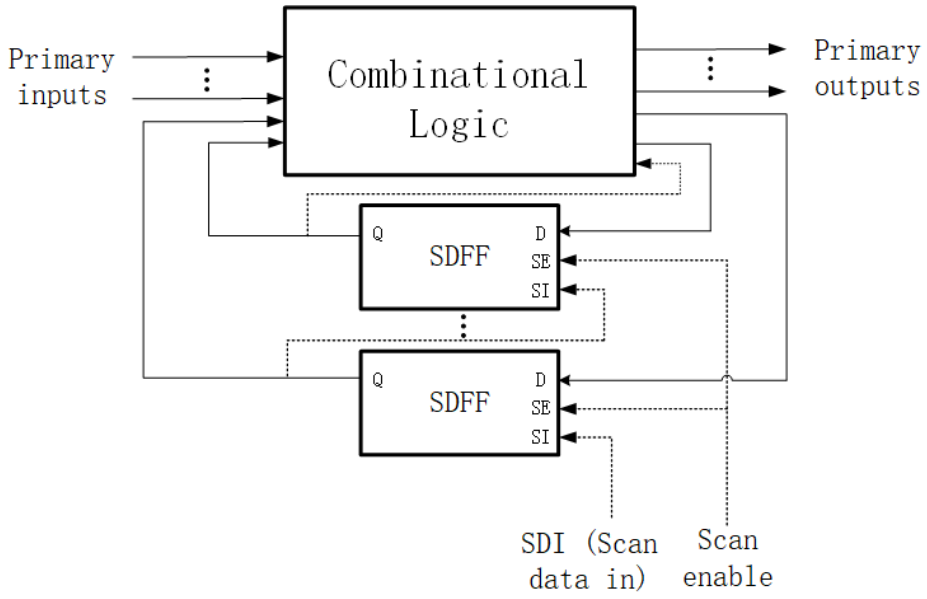

Figure 2.3: Scan design.

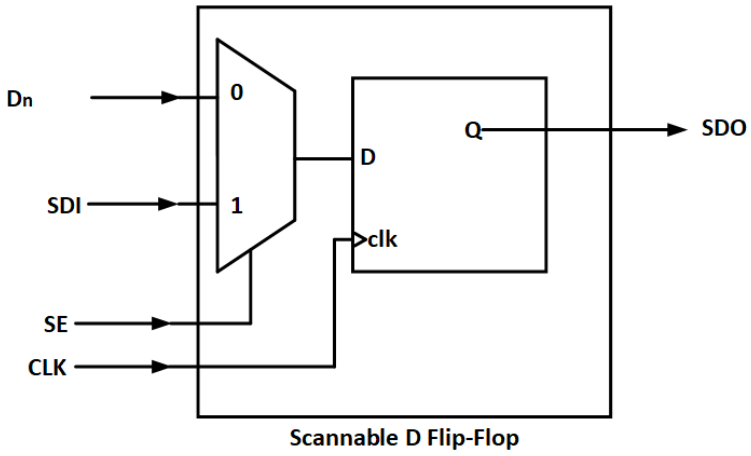

Figure 2.4: Scannable D-flip-flop.

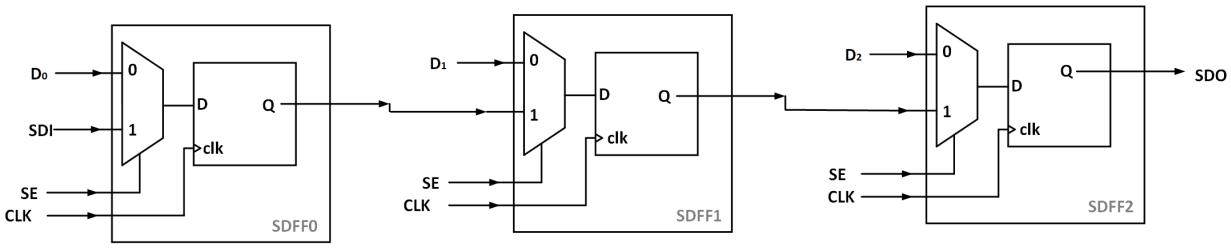

Figure 2.5: Scan chain

2.1.1.2 Scan Chain

If the scannable D-flip-flops are serially connected, they will form a shift register called a "scan chain," when **SE** is set to 1, as shown in Figure 2.5. (Note that the paths from the Q output of the flops to the circuit's functional logic are not shown in the figure.) In this scan chain, only 3 clock periods are needed to set the values of the flip-flops to values entered at **SDI** and to observe the values previously captured in the flip-flops at **SDO**.

2.1.1.3 Test Procedure

The procedure used during scan-based testing is conducted by applying a particular sequence of operations to the scannable D flip-flops over multiple clock cycles. The relationship of the different parts of the test procedure to each other in time are shown in Figure 2.6 and described in more detail below:

Shift in: When the clock is low, the scan enable signal, **SE**, is set to 1 to choose the test data path (either from **SDI** or its previous scannable D flip flop's **Q** output). **SE** is a global signal that changes the data source on all the scannable D flip-flops in the corresponding scan chains. Test data is applied to the pin that feeds the first **SDI** of the first scannable D flip-flop in the shift register. The circuit is now in shift mode and no longer operating as its original functionality. On each rising edge of the clock, a new value is shifted into the chain. The last shift-in clock edge will set all the scannable D flip-flops to the values needed by the test pattern. It is these values that will be used to test the internal logic of the circuit.

Capture: When the scan data is applied, right after the last shift clock cycle, the scan enable signal, SE, is set to zero to select the functional data path through each scan mux.

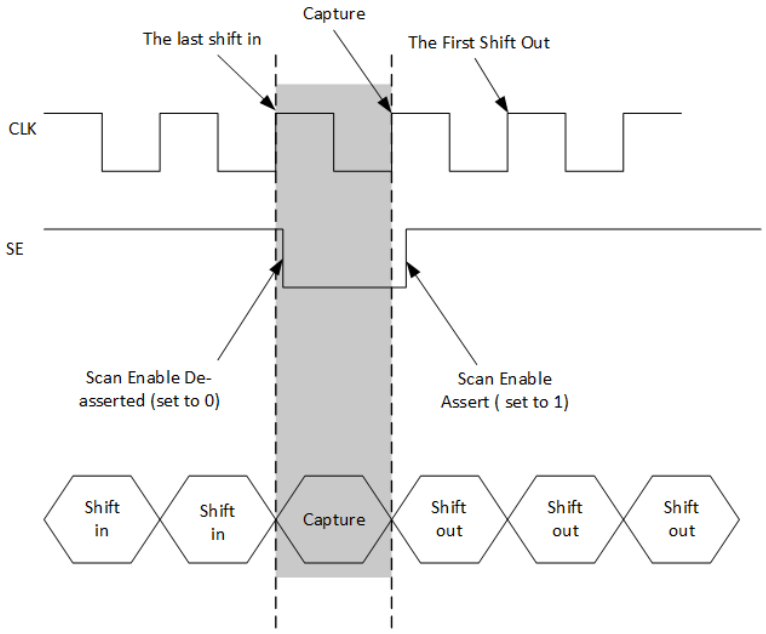

Figure 2.6: Scan Testing Procedure.

The functional data paths feeding the D inputs to the scan flip-flops should have the logic values associated with the circuit's natural response to the applied test pattern. On the next rising edge of the clock, the circuit's response, including any fault effect that reaches the flip-flops, is captured into the scan flip-flops.

Shift out: After capturing the circuit response, the scan enable (**SE**) signal is set to 1 again to select the scan/shift path (**SDI**) through the scan muxes to shift out the values in the scannable D flip-flops.

Once the responses are collected during shift out, and the final values of the output pins are collected, we can then compare the actual shift out values and output pin values with the expected shift out values and output pin values (expected values refer to the values that should be present when there is no defect in the circuit). If there is a discrepancy between the two sets of values, then we know that there is a problem with the circuit. (However, even if the values match, there could still be a defect that would cause an error for a different test pattern.) Unfortunately, the shifting process generally takes many clock cycles—all of which add to the total energy expended during test.

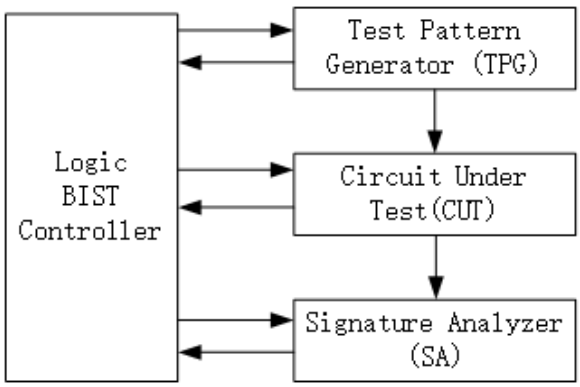

Figure 2.7: Overview of the BIST architecture [2]

2.1.2 Built in Self Test

Built-In Self-Test (BIST) [6] involves adding additional circuitry to allow a device or part of a device to perform a test without the help of an ATE. In general, adding BIST to a circuit involves the implementation of an on chip test pattern generator (TPG) and a signature analyzer (SA). Figure 2.7 shows a CUT (circuit under test) with BIST. When the circuit is being tested, the test pattern generator (TPG) generates patterns that are applied to the CUT, and a signature analyzer (SA) collects the CUT response to the test patterns and examines them. The signature analyzer has an expected output response to indicate if the circuit has passed or failed the test. Most BIST architectures that test the circuit logic (i.e. LBIST) use linear feedback shift registers (LFSRs) as a TPG because the LFSR can generate a sequence of good pseudorandom values with relatively little area overhead [7]. An LFSR is built using flip-flops and exclusive OR (XOR) or exclusive NOR (XNOR) gates. The signature analyzers (SAs) in the BIST architectures are commonly built from multiple-input signature registers (MISRs). A MISR (multiple-input signature register) also uses a version of an LFSR to collect and compact the test response after the test patterns have been applied to the circuit. Note that unknown values (e.g. generated from memories containing unknown data) entering the MISR will make the predicted signature indeterminate, and thus such values need to be avoided or masked. BIST is an especially good solution for testing of critical circuits that have no direct connections to external pins, such as embedded

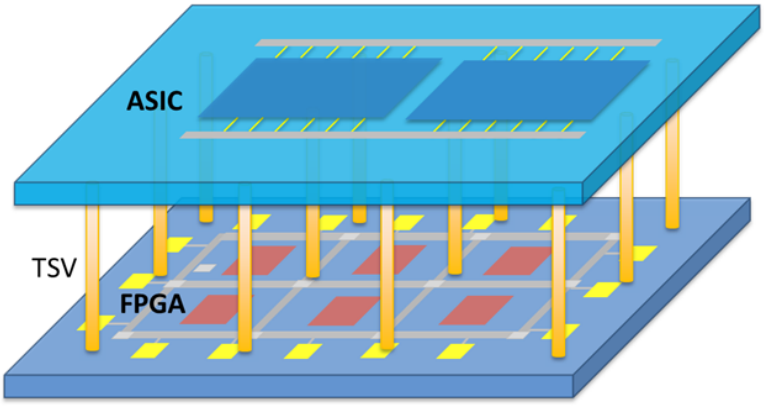

Figure 2.8: 3D stacked IC, which has shown before in the previous chapter

memories or logic cores used internally by the device [2]. However, especially in LBIST, obtaining 100% fault coverage is difficult with LBIST. Thus, deterministic top-off patterns may need to be added to help detect random-pattern resistant faults.

2.1.2.1 The Utilization of an FPGA for Test

As we mentioned earlier, using BIST is a good solution for testing a part of a circuit that has no directly connected external pins. In this subsection, we will briefly discuss the use of FPGAs in test as a BIST as well as the use of FPGAs on 2D boards and in 3D stacked ICs [8].

In a regular board, where the packaged chips are placed horizontally (this would be a 2D board), an FPGA may be used for several purposes. For example, it may replace a more expensive dedicated ASIC (Application Specific Integrated Circuit) to provide the required performance while meeting area or power constraints. In addition, the re-programmability of FPGAs allows designs to be modified easily over a system's lifetime, as specifications or standards change, or even as design errors or enhancements are discovered. Finally, 2D versions of FPGAs have been used for performance acceleration, allowing co-processing hardware to be reconfigured "on-the-fly" when a particular portion of the code can benefit [9]. It is reasonable to expect that these advantages of FPGAs will likely carry over into the 3D IC space.

An FPGA can be placed in one layer or multiple layers in a 3D stack and has the potential

to serve different purposes in a variety of applications. Intel has already created an embedded multi-die interconnect bridge (EMIB) used to connect its CPUs to Altera FPGAs to enhance performance and handle power issues [10]. Altera and Amkor have proposed a face-to-face packaging approach consisting of a mother die (FPGA) and daughter die (ASIC) [11]. Xilinx currently produces FPGAs that contains multiple FPGAs and other dies sitting side-by-side on a silicon interposer, aiding in prototyping and emulating large processor systems [12], [13].

FPGAs have also been used to aid in testing for many years. Boards may be partially tested (even when not all the chips and firmware are available yet), by adding more functionality to the load board at the factory or by connecting chips directly on the board [14]. When used to test other chips on the board, an FPGA can serve as a generator of tests for a directly connected chip. For example, memory built-in-self test (MBIST) is used to test memories and ultimately enable some repair of defective memories when enough spare rows and columns are available. If the MBIST design is programmed into an FPGA connected to a memory, the FPGA can be used to test the memory through a set of read and write operations. Alternatively, it may also serve as a target for functional or protocol-based tests—receiving information from or sending data to other chips based upon their functional behavior.

2.1.3 Test Data Compression

Because the amount of logic contained within digital circuits has increased due to circuit scaling, test time and test data volume have increased as well. Test data volume includes the data used to store the test patterns and responses. In a manufacturing environment, this information must be communicated to and from the ATE with a relatively small number of pins—leading to issues of test data bandwidth. To address these issues, various on chip decompressors and response compactors have been proposed to allow chips to be fully tested while minimizing the test data volume and test data bandwidth that are required (e.g. [3]). An example of this architecture is depicted in Figure 2.9. As shown in the figure, these on chip decompressors are used to feed many scan chains. This allows shorter chains to be used without increasing the required pin count, and thus it often helps to reduce the overall test time by reducing the number of shift cycles as well (provided that the chains are not made

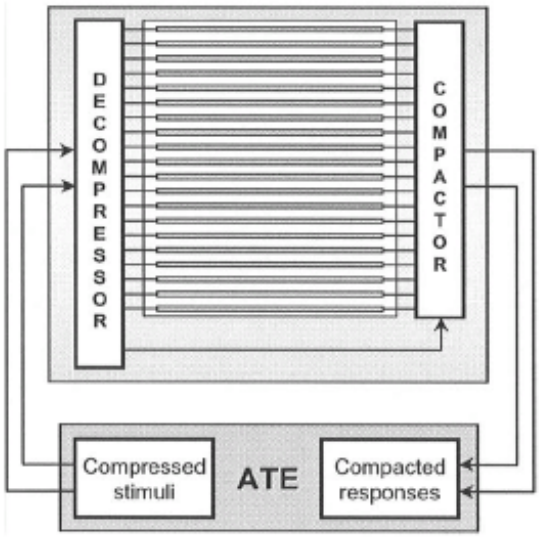

Figure 2.9: The overall architecture of Test Data Compression. [2]

extremely short).

The on chip decompressor (OCD) proposed in [3] is implemented by a ring generator and a phase shifter, as shown in Figure 2.10. The on chip decompressor design takes advantage of the fact that many don't care bits are present in deterministic test patterns that target a particular fault or set of faults. In other words, the faults will be detected regardless of the values to which those bits are set in the test pattern. Thus, the on chip decompressor takes a stream of information from its inputs and generates a stream of output values that feed into more scan chains than the number of decompressor inputs. An algorithm is used to determine what input stream is needed to guarantee that the needed deterministic bits will be present in the corresponding scan cells. The maximum compression provided by the approach corresponds to the ratio between the number of inputs to the ring generator and the number of scan chains (outputs from the phase shifter), provided that the scan chains are balanced.

2.2 Fault Models

Defects are physical problems that occur in an actual manufactured copy of a circuit. In contrast, faults are models of defects that predict how the presence of a defect will affect the

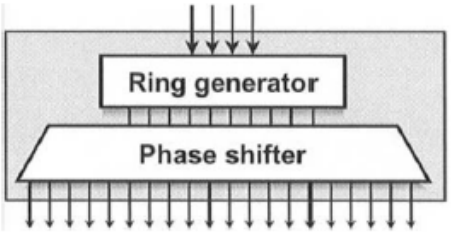

Figure 2.10: On Chip Decompressor [3]

logical behavior of that circuit. Fault models are used by ATPG algorithms to provide the targets for test pattern generation and to serve as a means of characterizing the effectiveness of the final test set. Thus, an ATPG tool will try to generate a test pattern set that will detect all of the faults in the fault list and achieve 100% fault coverage. In this dissertation, we concentrate on the two most common types of fault: the stuck-at fault and the transition fault.

2.2.1 Stuck-at Fault Model

The stuck-at fault model is the earliest and the most common fault model. A stuck-at fault [15] occurs when a wire in the circuit is permanently "stuck" at a fixed logic value. For example, this behavior could result from a short to ground or a short to power. Two conditions must be satisfied to detect a stuck-at fault:

- *Excitation*: The test pattern must ensure that the good and faulty circuits have different logic values at the fault site. This is achieved by justifying the appropriate logic value at the fault site in the good circuit to the appropriate bits in the test pattern.
- *Obvservation*: The difference in logic value between the good and faulty versions of the circuit must be propagated to a primary output or a scannable D flip-flop.

Even though all physical defects are not well-modeled by stuck-at faults—especially in circuits manufactured in today's advanced technology nodes—test sets that target stuck-at faults have historically been able to achieve good coverage of the actual defects. However, to

obtain even better defect coverage, other fault models have been explored. One of the most important of these additional fault models is the transition fault model.

2.2.2 Transition Fault Model

Some defects may cause gates in the CUT to have a higher than normal delay. This higher than normal delay means that the gate will switch at a lower speed than expected when the CUT's inputs change. This additional delay may cause changes in the output values of gates in the combinational logic of the circuit to not reach the outputs of the circuit or the inputs to the circuit's flip-flops within the needed clock period. When this occurs, the wrong values may be captured at the outputs or in the flip-flops, causing errors in circuit operation.

As a result, the transition-fault [16] model has been proposed to model a large amount of extra delay at a logic gate due to the presence of a defect. There are two types of faults possible in the transition fault model: a slow-to-rise fault and a slow-to-fall fault.

- A slow-to-rise fault refers to a slow transition at a gate input or output when the transition is from a logic 0 to a logic 1.
- A slow-to-fall fault refers to a slow transition at a gate input or output when the transition if from a logic 1 to a logic 0.

To detect a transition fault, a pair of test patterns is applied. The first pattern sets the value at the fault site to the appropriate value (logic 0 for a slow-to-rise fault and logic 1 for a slow-to-fall fault). The second pattern launches the transition by setting the value at the fault site to the opposite value in the good circuit (logic 1 for the slow-to-rise fault and logic 0 for the slow-to-fall fault.) In the faulty circuit, it is assumed that the transition doesn't happen and that the fault site remains at the initial logic value for a long time.

The second pattern must also propagate the value at the fault site to an observation point, which could either be a primary output or a scannable D flip flop. Because the delay is assumed to be large, the length of the path to the output/flip-flop is assumed to be irrelevant. Depending on how the transition is launched and captured, there are two transition fault pattern generation and application approaches used in scan-based circuits: launch-off-shift [17], launch-off-capture [18].

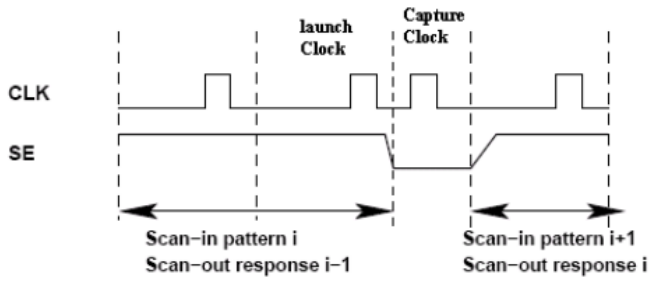

Figure 2.11: Waveform for Launch-off-Shift delay test [2].

2.2.2.1 Launch-off-Shift Method (LOS)

In the launch-off-shift (LOS) approach [17], the transition at the gate is launched by the last shift cycle during the shift operation, as shown in Figure 2.11. The launch clock is then immediately followed by a fast capture clock pulse such that the time difference between launch and capture corresponds to the clock period at which the circuit is being tested. The scan enable (**SE**) signal is high during the last shift and must go low within one system clock period to allow capture at the capture clock pulse. So the **SE** signal, which typically drives all scannable D flip-flops in the CUT, should also switch to low for all scannable D flip-flops within the specified time for capture to occur at system clock speed. This requires the **SE** signal to be driven by a well-designed buffer tree or strong clock buffer, and it is costly to implement.

2.2.2.2 Launch-off-Capture Method (LOC)

In the launch-off-capture approach [18], the circuit's logic itself is used to launch the transition at the targeted gate. As shown in Figure 2.12, after the last shift cycle, the first capture-mode clock cycle is applied to the CUT (launch clock). This launches the transition between the first and second pattern. Then, one more capture clock is used to capture the test results. Only the time between the two capture pulses must be "at-speed." As a result, the **SE** signal is not required to de-assert at-speed to initiate a transition at the gate.

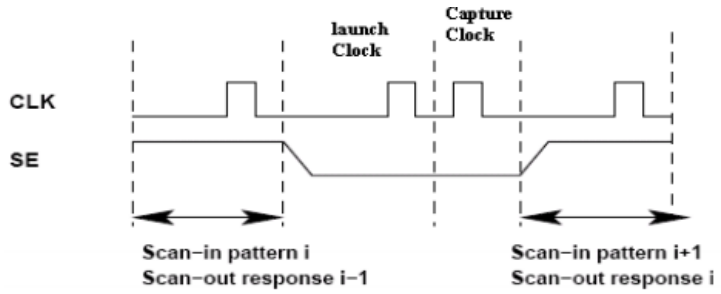

Figure 2.12: Waveform for Launch-off-Capture delay test [2].

2.3 Test Power Issues

Excess power during test can increase circuit delays, cause IR drop, cause overheating, and damage or reduce the long term reliability of a chip [19]. Achieving low test power may be especially important when tests need to be applied in the field—such as in an environment where other external factors are already leading to increased temperature. Thus, effective techniques to minimize power expended during test are needed [20]. Minimizing power consumption in VLSI circuits increases the life expectancy and the reliability of the circuit [21], [22]. The estimation of the power consumption during test is based on the following equation 2.1 [23].

$$p = \frac{1}{2} f \cdot V_{dd}^2 \cdot \sum_g C_g \cdot N_g \tag{2.1}$$

Here f, V_{dd} and C_g denote clock frequency, supply voltage and capacitance of gate g, respectively. N_g denotes switching activity, i.e., the number of gate output transitions per clock cycle. If we consider the frequency, voltage, and capacitance to be pre-determined, then optimizing the the power consumption during test will be solely related to the number of transitions that occur during test.

As mentioned earlier in Section 2.1.1.3, the power consumption can be divided into two stages during test: shift and capture. Together with Equation 2.1, we can deduce that we need to reduce the number of transitions during test in either or both of the stages to be able to effectively reduce power consumption.

Historically, multiple researchers have previously attempted to reduce test power during both shift and capture based on Equation 2.1 by using a variety of test generation and DFT techniques. The techniques are elaborated as follows.

Common techniques to reduce power during scan shift include adjacent fill [4] and test vector reordering [24]. Some researchers have attempted to prevent data from flowing into the combinational logic during scan shift by preventing the flow of changing logic values into the combinational circuitry during shift [25], [26], [27]. For example, the authors of [25] removed power to the first level of logic after the DFFs during the shift procedure. Reordering of the scan cells can also be used to reduce power during shift. For example, in [28], scan cells with similar weights during weighted random scan were grouped together into the same chain to help reduce shift power.

Other researchers have focused on reducing scan shift power in the presence of an on-chip decompressor (e.g. [29], [30], [31], [32], [33] and [34]). For example, the authors of [29] modified the decompressor hardware to allow a constant value of 0 to be shifted into a group of chains over multiple shift cycles, with those shift cycles determined by the values in a control register. A related approach was taken in [30], and it was noted that scan chains that only load constant values could be kept in shift mode when the test is applied to reduce the power of scan chain unload as well.

Capture power reduction has also been identified as being important, especially in the context of at-speed test. Some researchers have attempted to reduce this capture power by intelligently filling don't cares in the test patterns to reduce power draw during the capture cycle (e.g. [35], [36]). In some cases, instead of simply lowering power, an attempt is made to create pseudo-functional patterns that are more similar to the those seen during normal operation (e.g. [36], [37]). The authors of [36] applied this concept to an on-chip decompressor, where they allow the shifted out data to be fed back to the beginning of the chain so that the data shifted in can come from either the decompressor or the value that has been shifted out. When applying an X-filling encoding algorithm, the authors of [38] used the existing MISR data to select the possible flip flops to be enabled during capture; [39] and [40] further modified the EDT decompressor by adding encoded blocks and shadow registers. However, the reduced number of don't cares that result from X-filling approaches

can lead to reduced efficiency of the test pattern compression algorithms.

Disabling flip-flops from capturing to reduce capture power has also been discussed. In [41], additional test points were added to the already-present clock-gating circuitry to allow different clock gating circuitry to be independently controlled. The ATPG procedure was then modified to take advantage of these additional test points—reducing test power at the cost of additional pattern count. The authors of [42] also attempted to reduce capture power through modifying the test generation procedure to reduce the number of flip-flops to which fault effects propagate, and they then cluster and reorder the flip-flops into chains based upon the optimized test patterns. A limitation of these approaches is that they both require changes to the test generation procedure. Disabling individual chains or groups of scan chains has been proposed as well (e.g. [43], [44]). For example, the authors of [44] have created a low-power BIST architecture that breaks the scan chains into groups that can be disabled together, allowing constant values to be supplied to sub-circuits of the combinational logic. In [45], [46] and [47], the scan chains were divided into shorter chain segments that could be shifted independently, allowing fewer simultaneous transitions during scan shift. In [46], only one chain was allowed to capture a test response at given time, further reducing overall power. The work of [46] was extended in [43] to change the test generation procedure to create special test patterns targeted toward their architecture.

In the next chapters, three architectures that complement this previous work in low power test will be introduced.

Chapter 3

Reducing Capture Power through the Use of a Control Chain

The work in this chapter was first presented in [48].

3.1 DFT Architecture for Segment-based Capture Control

As mentioned earlier, our approach complements the previous work in the realm of low-power test pattern generation. It takes a fine-grained approach to preventing particular segments of the chain from capturing data when those scan segments are not needed by the current pattern for fault detection. Furthermore, we aimed to do so in a way that is *i*) simple to implement, *ii*) scalable, and *iii*) valid even in the presence of an on-chip decompressor. Here, we will describe a solution that:

1. Does not require changes in the test generation procedure carried out by a modern commercial ATPG and fault simulation tool.

2. Does not force the DFT insertion process to be dependent on a particular test pattern set.

3. Retains the original test pattern set as well as the original fault coverage with no increase in test pattern count.

4. Does not require internal changes to an on-chip decompressor.

To satisfy the stated requirements, here we describe a solution that adds one or more "control chains" to a design. Each functional chain is broken up into multiple segments—each of which is controlled by a bit in the control chain. That bit determines whether or not the segment will capture data during the capture clock cycle. If targeted faults will be detected in that segment, then capture is enabled. Otherwise, the segment will hold its value. This will not only prevent the flip-flops in the segment from changing value, but it will prevent

any transitions that would have arisen from those flip-flop toggles from propagating into the rest of the circuit.

For example, to see how the proposed solution can be implemented, consider Figure 3.1. In this figure, Mux-D scan flip-flops for the functional chain are shaded in grey, and the Mux-D scan flip-flops for the control chain are shaded in green. The functional chain is broken up into two segments: the first segment in the figure is only one bit long while the second segment (to the right) is two bits long.

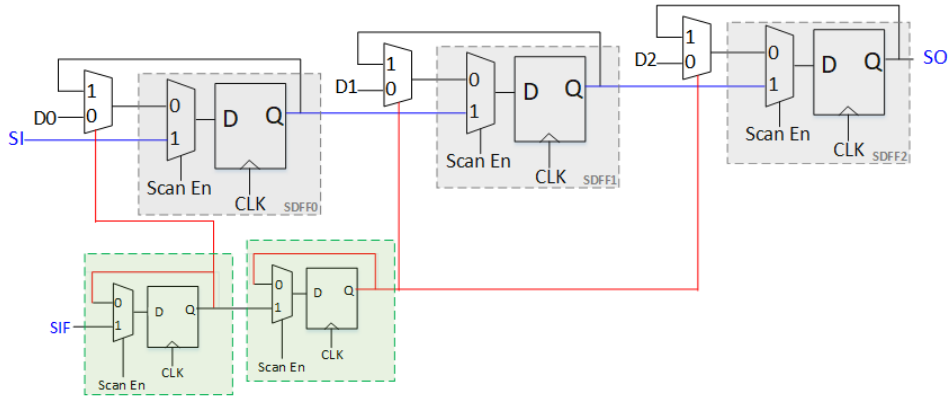

Figure 3.1: Schematic design of segmented scan chain.

The value in each control flip-flop determines whether the segment it controls will capture a new value or maintain its old value. There are multiple ways to implement this; however, in the figure, we have chosen to insert an additional multiplexer before each scan flip-flop. The select line for this multiplexer is connected to the control flip-flop for that segment. When the control value is equal to a logic 1, the functional flop will hold its original value on the next capture cycle because the Q value at the output of the flip-flop will be fed back into the input. In contrast, if the value in the control flip-flop is equal to zero, the functional flip-flop will capture data from the circuit itself (shown as D0, D1, D2, etc.) provided that the scan enable signal, *Scan En* is set to zero, and thus the chain is not in shift mode.

If the approach is implemented directly as seen in the figure, then there is some additional delay in the functional path due to the presence of the additional mux. If only a few flip-flops are destinations of critical paths, then this could be handled by removing the mux from those flip-flops and not disabling them. Alternatively, other approaches that disable the flip-flop by gating the clock or adding an enable input, etc. could be used instead.

In the figure, the control chain is fed directly from an additional scan data input. This allows the control chain to be implemented independently of any on-chip decompressor that may be present, although one could potentially generate the control bits with the on-chip decompressor if one designed the control chain appropriately and had access to the on-chip decompressor ATPG tool's algorithm.

Note that we expect that the control chain may be shorter than a normal scan chain even if it controls segments on multiple chains. Thus, only a single additional input may be needed for control if the number of segments is not too large. It could also be possible to use a de-multiplexer at this input to shift data into multiple short control chains in-turn. This could potentially allow those control chains to be located closer to the segments they control and reduce the delay between the control chain and segments. A detailed analysis of the advantages and disadvantages of such an approach is left to future work.

Note that this implementation was designed with the intention of handling static test patterns. Minor modifications should allow our approach to work with dynamic patterns implementing both launch-off-shift and launch-off-capture.

3.2 Methodology

To implement the proposed approach, it is necessary to a) split each scan chain into multiple segments that can be independently controlled during capture, and b) determine which segments should have capture enabled for each pattern. Intuitively, if a segment only detects faults on the current pattern that have already been detected by a previous pattern, then that segment can be disabled for the current pattern. Once the segments that should be enabled for each pattern are determined, they can be used to generate the control values that will ultimately be shifted into the control chain.

In this architecture, data were collected for four different circuits obtained from open-

cores.org. The characteristics of these circuits are listed in Table 4.2.

Table 3.1: Benchmark circuit properties

Circuit	# of scan DFFs	# of scan chains	avg. length of scan chains	# of patterns
des56	382	4	96	114
fm_receiver	521	5	104	388
colorconv	858	9	96	150
fpu_double	5434	5	1087	476

For each circuit, the following procedure is followed:

1. Insert the scan chains.
2. Create the on-chip decompressor logic using a commercial software tool.
3. Run ATPG and record the patterns being shifted out of the on-chip decompressor into the scan chains.
4. Determine which faults are detected by each flip-flop in each pattern.
5. Divide the scan chains into segments of equal (or approximately equal) length.
6. Iterate through all patterns.
   (a) Identify which faults have been detected by a previous pattern and remove those faults from consideration.
   (b) Consider each segment in turn. If the segment detects new faults, record that fault as newly detected and specify that capture should be enabled for this segment and pattern. Otherwise, disable capture for this segment on this pattern.
   (c) For the segments that are currently still considered enabled on this pattern, we apply the procedure shown in Figure 3.2 to further disable some of these segments.

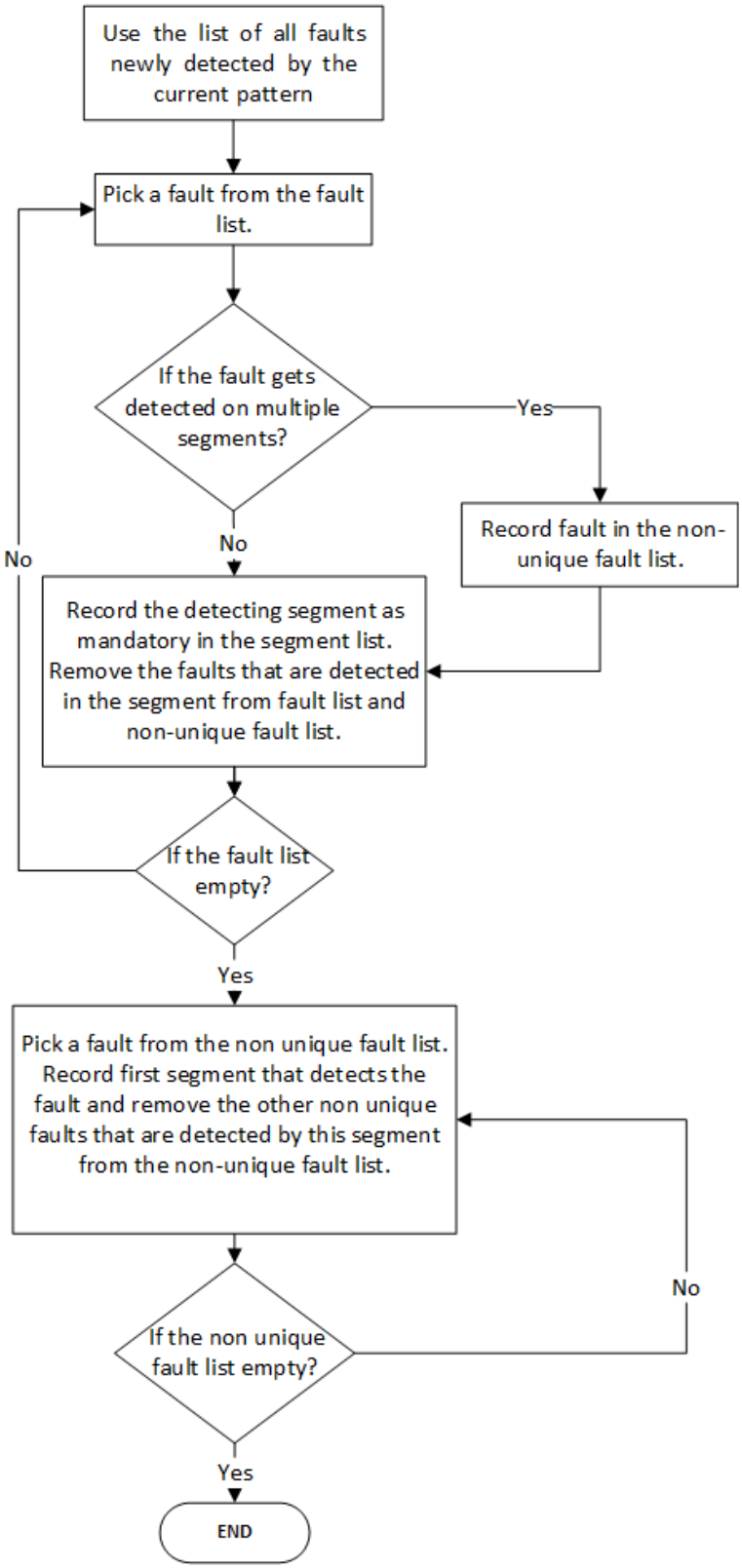

Figure 3.2: Procedure to determine which segment to keep when faults are detected in multiple segments.

The procedure shown in Figure 3.2 is an approach that allows us to identify faults that are detected by multiple segments so that only one of the multiple segments will have capture enabled—provided that the other segments are not needed for the primary detection of another fault. A more detailed description of this procedure is shown below:

1. Use the list of all faults newly detected by the current pattern in part (b) above.
2. Iterate through all of the faults in the list.
   (a) If the current fault is detected by only one segment, we call it a *unique* fault. The segment that detects the *unique* fault is added to the list of mandatory segments for that pattern, and all other faults detected by that segment are removed from the fault list.
   (b) If the current fault is detected by more than one segment, it is added to the *non-unique* fault list. (Note that when a segment is identified as mandatory, both faults that have not been considered yet as well as faults that are in the non-unique fault list will be dropped if they are detected by that mandatory segment.)
3. Iterate through all of the remaining faults in the *non-unique* fault list.
   (a) Add the first segment that detects this non-unique fault to the enabled segment list. (This list was previously composed only of the *mandatory* segments.)
   (b) Remove all faults from the non-unique fault list that were also detected by the selected segment.

Once the entire process is complete, we know which segments should be enabled during capture for every pattern to maintain the fault coverage of the original test set. This information can then be recorded as control bit data that will be shifted into the control chain for each pattern.

3.3 Results

As already noted, to evaluate the effectiveness of our approach, we ran our experiments on circuits obtained from opencores.org. A commercial tool was used to insert scan chains in each circuit, create an on-chip decompressor, and generate a low power test pattern set.

For each circuit, the steps outlined in Section 3.2 were followed to identify which segments should capture data for each test pattern in the low power stuck-at test pattern set. The goal was to see how much *additional* capture power reduction above and beyond that which was already obtained by the commercial tool is possible with this approach. Because the size of the segments affects the degree to which fine-grained control and power reduction can be achieved, the analysis was repeated for different segment lengths.

Different segment lengths correspond to different amounts of overhead. In particular, each segment requires an additional control flip-flop to be inserted into the control bit shift register (i.e., the green flip-flops in Figure 3.1). In this chapter, we characterize this overhead as a percentage of the original number of flip-flops in the circuit. Thus, a chain with 10 segments and 100 flip-flops would correspond to an overhead of 10%.

We estimate the capture power by counting the number of flip-flops that toggle during capture for all of the segments for which capture is enabled. We did not simulate the effect of this toggling on the combinational logic for this set of experiments. However, if those simulations were performed, the overall reduction in circuit switching would be even greater than that shown below. The toggling reduction is calculated as:

$$Toggle\ reduction = \frac{Toggles_{disabled}}{Toggles_{all}} * 100\% \tag{3.1}$$

where $Toggles_{disabled}$ refers to how many toggles would have happened in the disabled segments and $Toggles_{all}$ refers to the toggles that occur when all the segments capture.

3.4 Capture Power Reduction Based on Forward Simulation

Figure 3.3 shows the plot of average percent reduction in toggle count as the overhead is increased (i.e., number of segments per chain is increased). Experiments for each circuit were run until the overhead exceeded 10%.

For all four circuits, increasing the overhead (i.e. using shorter segments and more control bits) reduces the overall toggle count—thereby reducing capture power. For smaller circuits, we see that the toggle reduction at approximately 3% scan chain overhead is about 60% for *des56*, 60% for *fm_receiver* and 70% for *colorconv*. For our largest circuit, *fpu_double*, we see that with an overhead of approximately 3%, we can achieve almost 90% toggle reduction

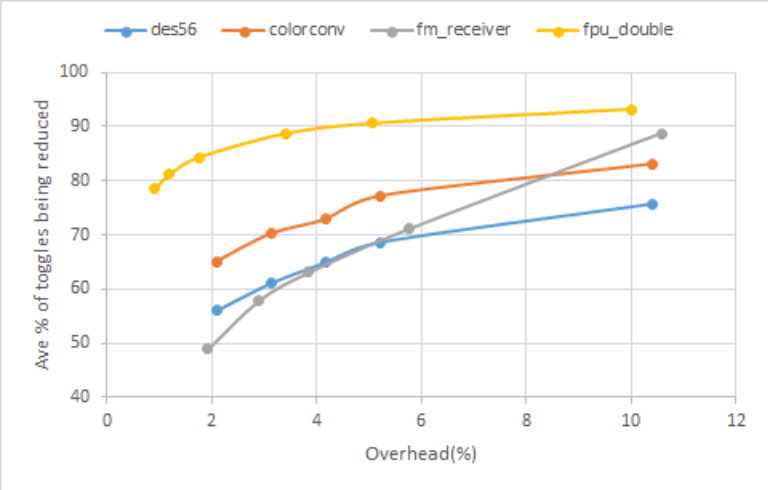

Figure 3.3: Average % reduction in toggle count.

on average. Increasing the overhead allowance to approximately 10% allows us to achieve an average toggle reduction of almost 93%.

Note that increasing the overhead by decreasing the segment length allows much better toggling reduction for all circuits. This makes sense because it is easier to pinpoint a small number of flip-flops that will be used to achieve the desired fault detections when the segments are smaller. In the limit, each flip-flop could be its own segment, although the overheard would be prohibitive. Fortunately, the data shows that high toggle reduction can be achieved with much lower overhead.

Also note that the proposed approach is especially effective for the *fpu_double* circuit. It is not only the largest circuit we studied, but it also contains many don't cares in its test patterns. Many of those test patterns are required to detect just a few new hard-to-detect faults. As a result, it may be that only a single segment may need to remain enabled for detection for many test patterns. This is different from a circuit such as *des56*, which is more observable, requires fewer patterns, and tends to detect more faults per pattern. As a result, *des56* is more likely to have more segments enabled on each pattern.

Figures 3.4–3.6 give more detailed data for *fpu_double* for an overhead of approximately 3.4%. The data for other overheads and other circuits follow similar trends.

Figure 3.4 shows the percentage of segments disabled during capture as a function of

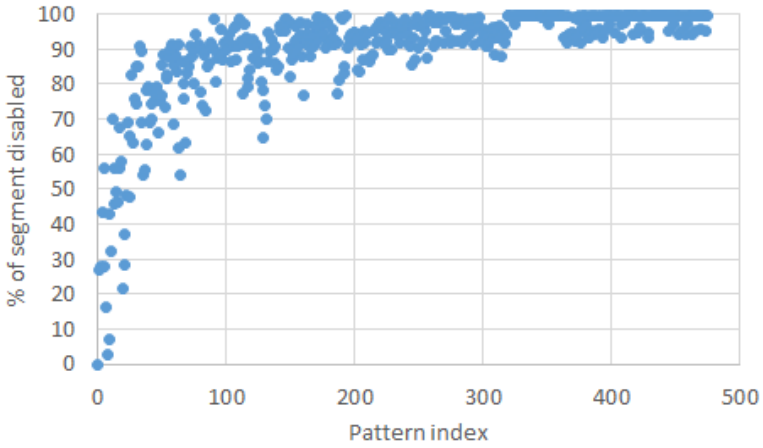

Figure 3.4: Percentage of segments disabled during capture for *fpu_double* circuit with overhead of 3.4%

pattern index. Here, the pattern index refers to the number of patterns in the test set that have been previously been applied during testings. We see that as testing progresses, and the pattern index increases, the percentage of segments that can be disabled starts to increase rapidly. Our experiments show that for 113 patterns in the overall pattern set (i.e. 23.7% of the patterns) all but *one* segment can be disabled. The corresponding toggle reduction for each pattern is shown in Figure 3.5 for the same circuit and overhead. We found that the average toggle reduction was 90% and the median reduction was 93.3%. In some cases, the toggle reduction is almost 100%. As shown in Figure 3.6, we also see a clear correlation between segments disabled and toggle reduction.

As mentioned earlier, *fpu_double* was partitioned into 5 scan chains. We also wanted to determine whether some chains had significantly more toggles than others. The total number of toggles during test for each chain are shown in Table 3.2 under the "original toggle" column. The column to the right shows the number of toggles that can be *removed* from each chain during test across all test patterns. The last column shows the percentage of the original toggling that we were able to remove with the proposed approach. We see that the toggle reduction is distributed fairly evenly across the five chains, although the last chain has less percent reduction and more toggles overall. However, that chain also had

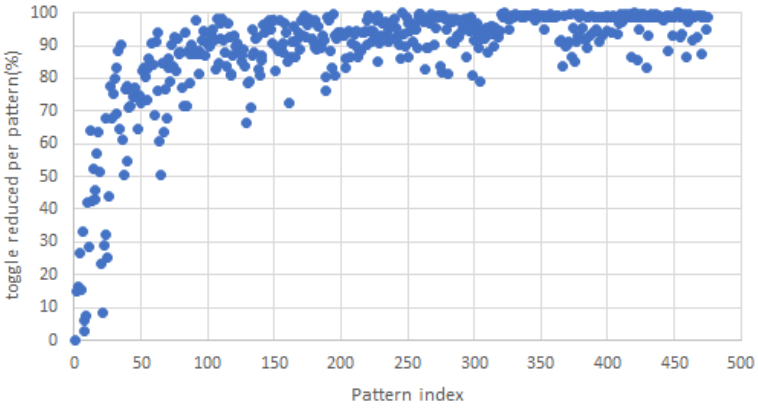

Figure 3.5: Toggle reduction per pattern during capture for *fpu_ double* circuit with overhead of 3.4%

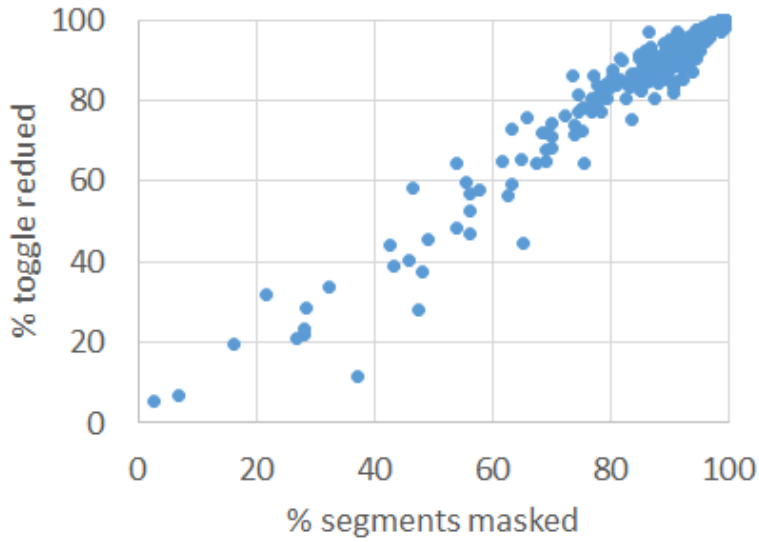

Figure 3.6: Correlation between toggle reduction and segments disabled.

more toggles originally. If additional reduction were needed, reordering and/or distributing flip-flops among the chains could be a possibility.

Table 3.2: Toggle count distribution across the 5 chains in *fpu_ double*.

chain number	original toggle	reduced toggle	% reduction
1	145244	131877	90
2	202195	183840	90
3	189093	171303	90
4	189093	143222	92
5	313673	259479	82

3.5 Additional Capture Power Reduction based on Reverse Simulation

Implementing reverse simulation is a well-known technique to reduce the size of a test set. Intuitively, faults detected at the end of ATPG are "hard" faults that must be deterministically targeted. In contrast, some of the faults targeted early during test pattern generation may be "easy" faults that would have been detected fortuitously by a later pattern anyway.

In our capture power reduction approach, we note that in the early patterns, all or almost all scan segments are needed for capture because we only disable those segments that are not capable of detecting any new faults. This behavior is clearly shown in Figure 3.4. Intuitively, all of the segments will be retained for the first pattern because they are *all* capable of detecting a new fault (e.g., a stuck-at fault on the D-input to the flip-flop if nothing else).

Thus, in this section, we wanted to see if any additional capture power reduction was possible through a type of reverse simulation. Specifically, we start with all segments that are enabled for each pattern after forward simulation has been completed. We then consider the patterns in reverse order and determine which faults are detected by each pattern on the enabled segments. If any enabled segment does not detect a new fault when the simulation is considered in this reverse pattern order, then it can be removed as well. Figure 3.7 shows the

detailed algorithm for this approach. $Fault\_List\_Global$ in the algorithm refers to faults that are detected by the enabled segments in all patterns considered in reverse thus far; $Fault\_list\_i\_j$ refers to faults that are detected by enabled segment j in the i^{th} pattern.

Figure 3.8 compares the toggle reduction of *fpu_double* with an overhead of approximately 3% with and without backwards simulation. It is clear that additional reduction is possible. In fact, for some specific patterns, up to an additional 13% reduction was seen. Such patterns are closer to the beginning of the test set because those are the patterns that showed the least reduction originally, and by the time we reach those patterns through backward fault simulation, many easy faults would have been detected already. Thus, with very little additional effort, we can disable capture for even more segments.

3.6 Area overhead and delay

After the control chain has been added, the global routing might be an issue because the wire spans across the whole circuit. The timing delay might also be an issue; the added muxes could add extra delay in the functional path. We extracted the area overhead and delay results from a synthesis tool. The results are presented in Table 3.3.

Table 3.3: Area Overhead and Timing Delay with extra control chain added

circuit name	# of scan chains	segment length	original area	area overhead %	original timing delay	timing increased percentage%
des56	4	30	20527	13.1	2.19	5.9
colorconv	9	30	77533	10.72	47.98	1.5
fm_receiver	5	30	42047	16.72	30.25	2.45
fpu_double	5	30	535049	15.0	96.74	0.48

If the length of the control chain becomes too long, multiple control chains can be added. However, each additional chain will require a new input pin.

3.7 Conclusion

In this chapter, we have shown that very high toggling reduction can be achieved during capture cycles even when we start with patterns that have been created by a commercial tool to achieve low power. Note that because we are disabling capture in scan chain segments, any effort made by the ATPG algorithm to reduce shift power for those segments for the pattern shifted into the chain will perform a "double duty". A reduced number of transitions during scan in will lead to a reduced number of transitions during scan out for the bits in those disabled segments because the values will be identical during shift in and shift out.

Because good results can be achieved with low flip-flop overhead, it should often be possible to completely shift data into a single control register that controls multiple functional chains in the same time required for a functional chain's load/unload operation. In addition, because toggling in the control chain will not feed into the circuit's combinational logic, it should have a minimal impact on power compared to toggling in the normal functional chains.

Also note that our approach tries to maximize the toggling reduction using a very greedy approach. The overall reduction is especially high for the largest circuit, and it reaches approximately 90% on average with little effort and overhead. We can get even more reduction (up to 13% for specific patterns that originally still had high toggle counts) when we remove additional segments by simulating the patterns again in reverse order. However, the final patterns (i.e., those generated last during ATPG) still have much less toggling than the initial patterns (i.e., those generated first during ATPG).

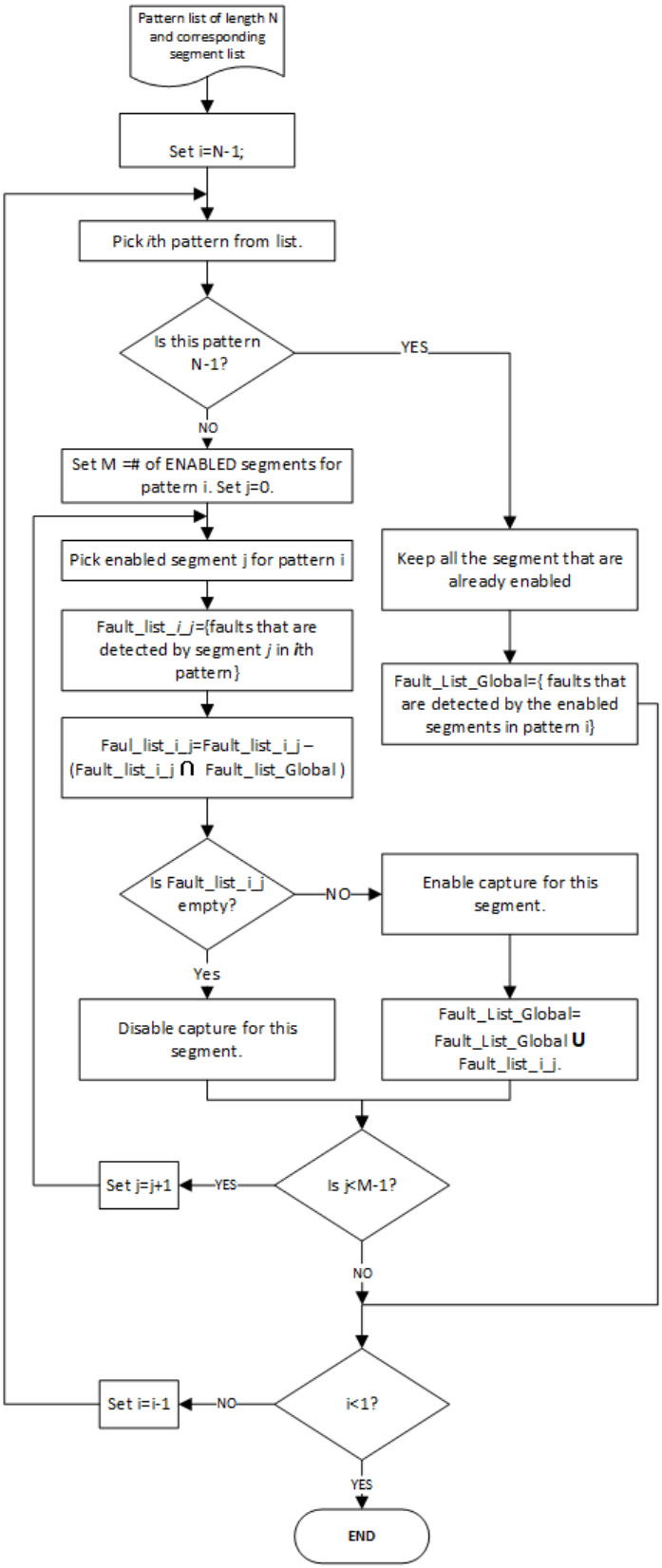

Figure 3.7: Reverse Simulation Flow

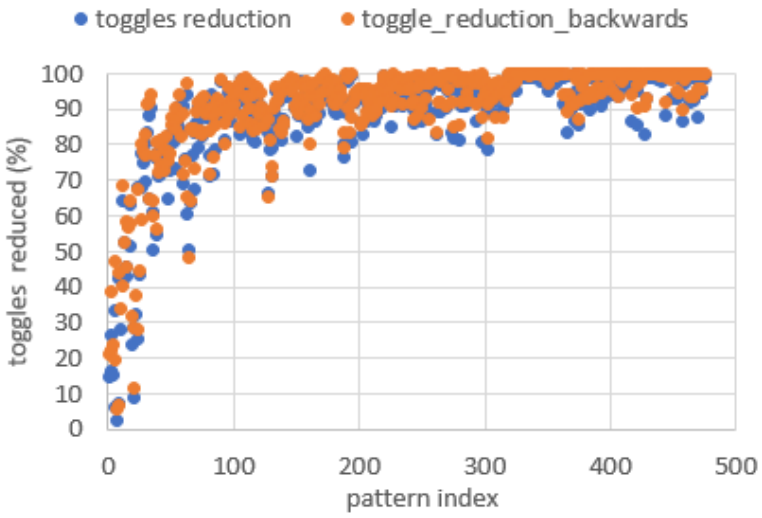

Figure 3.8: Comparison of toggle reduction between original flow and Backwards Simulation flow.

Chapter 4

Low Power Test through ATPG-Configured Embedded Enable Capture Bits

Work contained in this chapter has been accepted for publication at the International Test Conference 2021. [49]

Although the first architecture can achieve great test power reduction, a large amount of post processing was required to determine appropriate values for the control bits. Thus, in this chapter, we explore an alternative approach that removes the need for the extensive post processing by embedding the segment control bits within the chains themselves. This also removes the need for the expensive extra pin that is required for the first architecture.

Like the first architecture, this second approach is compatible with an on-chip decompressor. However, while the on-chip decompressor was bypassed when implementing the control chain in the previous chapter, in the second architecture, the control bits will be fed by the on-chip decompressor itself. We will also discuss in this chapter, that when inserted into a full-scan design that already contains flip-flops with an enable input (such as for clock gating), no additional logic must be inserted in the paths of the functional logic—minimizing the impact on the functional circuit delay.

4.1 Second DFT Architecture for Segment-Based Capture Control

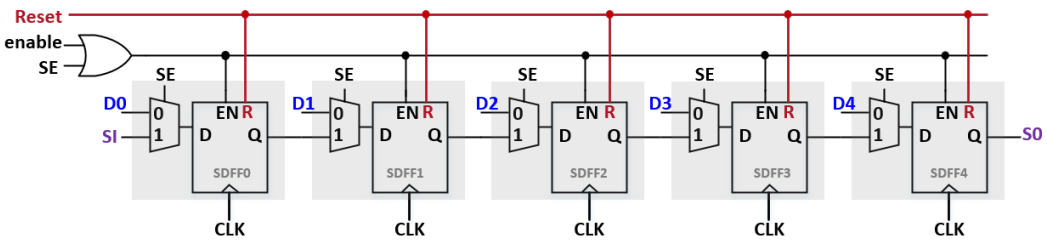

Figure 4.1: Schematic design of regular scan chain with enable signal.

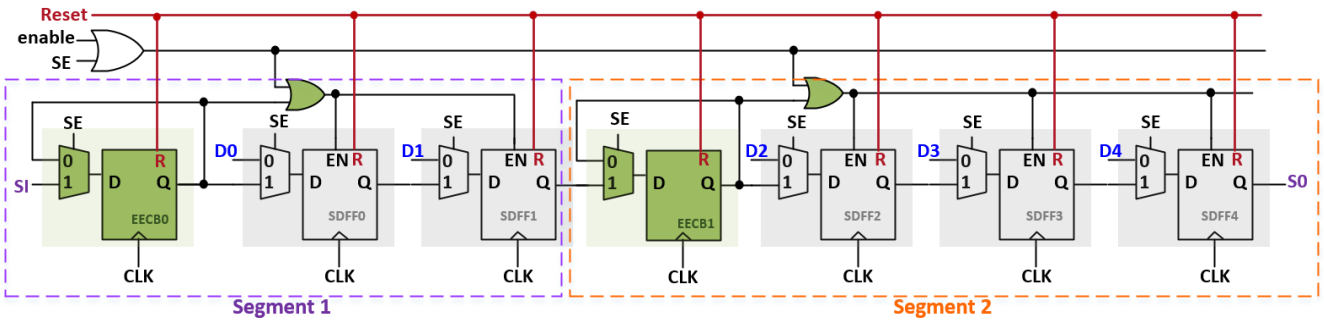

Figure 4.2: Schematic design of segmented scan chain.

Figure 4.1 shows a scan chain consisting of five scannable D flip-flops. In this example we assume that the scannable D flip-flops (SDFF0–SDFF4) consist of a D flip-flop with a multiplexer added at the input—forming a MUX-D scannable D flip-flop outlined in grey. One input of the MUX acts as the functional input (labeled D0–D4), and the other input serves as the *Scan In* (SI) input. A *Scan Enable* (SE) signal is used as the multiplexer select line. These scannable D flip-flops also have an enable input (EN). Each flip-flop will only clock the data in when the EN input is asserted. A global *enable* signal dictates whether the flip-flops capture data from combinational logic that feeds into D0–D4 (not shown) during normal functional operation. De-asserting the EN signal prevents the change of data in the flip-flops. The Scan-Enable signal (SE) is OR'ed with the global enable to ensure that data can be shifted into the flip-flops during test regardless of the value of the global enable. However, the global enable signal must be asserted during the capture cycles of test when SE is de-asserted. The global enable signal can be used during functional operation of the circuit to allow this part of the circuit to enter a low-power mode by preventing changes in the flip-flop values.

Figure 4.2 shows the same five flip-flop scan chain (shaded in grey) from Figure 4.1 split into two segments using two additional EECB flip-flops (shown in green). Segment 1 consists of two scan cells SDFF0 and SDFF1. Similarly, segment 2 consists of three scan cells SDFF2, SDFF3, and SDFF4. In this figure, each EECB scannable D flip-flop is connected to the first scan-flop of the segment it controls. Unlike the circuit shown in Figure 4.1, here the global *enable* signal is set to low during test. This will allow the EECB cells to solely

control whether or not the scan chain captures data during test. When SE is set to *high*, the segmented scan-chain works as a regular shift register. When SE is set to *low*, the outputs of the OR gates (shaded in green) are determined by the outputs of EECB bits. The value in each EECB flip-flop determines whether the segment it controls will capture a new value or maintain its old value. A logic 0 stored in the EECB flop at the start of a segment leads to a 0 at the output of the OR gate connected to the *EN* pin of the flip-flops in that segment. This has the effect of denying capture and the segment flip-flops retaining their previous value. In the example of Figure 4.2, using two EECB flops, we can control the two segments independently and have both segments disabled, both segments enabled or only one of the two segments enabled. In functional mode, the EECBs should be set to 0 to allow the global *enable* to have full control of whether or not the functional flip-flops capture data.

In this example, our approach harnesses the existing *enable* inputs to the flip-flops in the circuit as seen in Figure 4.2 to allow or deny capture. Alternatively, other approaches, such as disabling the flip-flop by gating the clock, could be used instead to disable capture during test. The hardware overhead compared to the original chain of Figure 4.1 is one MUX-D flip-flop and one OR gate per segment, plus the additional routing occurring locally.

In Figure 4.2, the EECB bits become part of the original scan chain. This allows the ATPG tool to generate the appropriate logic values for the EECB bits to capture data in those segments that are needed to detect targeted faults for a particular test pattern. Even though this approach only appears to save on capture power, we will show with an example below that it can also save on shift.

In Figure 4.3, we have 21 bits in the scan chain ranging from SDFF0 to SDFF20. The scan chain is shifting values from left to right. Assume the initial state of the values in SDFF0–SDFF20 is **001110101110100101100**, and the new pattern generated by the ATPG tool that will be shifted in is **001XX11X010X01X01XX01**. The don't care bits in the new pattern are randomly filled with zeros or ones, and the new pattern that is actually shifted in is **001101110100011011101**.

All the values in the scan cells (SDFF0 – SDFF20) after the first shift cycle are displayed in the third row in Figure 4.3. If a toggle occurs in a SDFF, then the color of the box for that SDFF is changed to a light grey. We can see that the total number of toggles after the

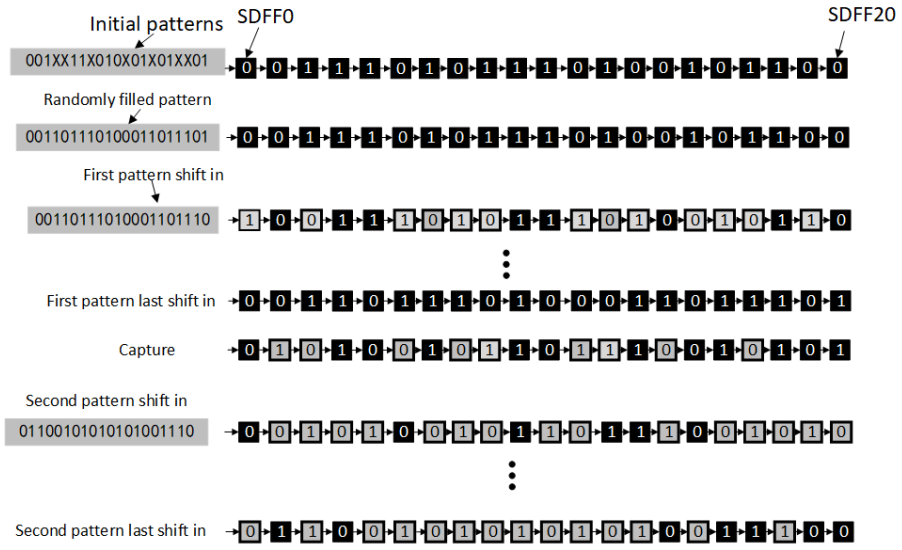

Figure 4.3: Shift in random patterns into a regular scan chain.

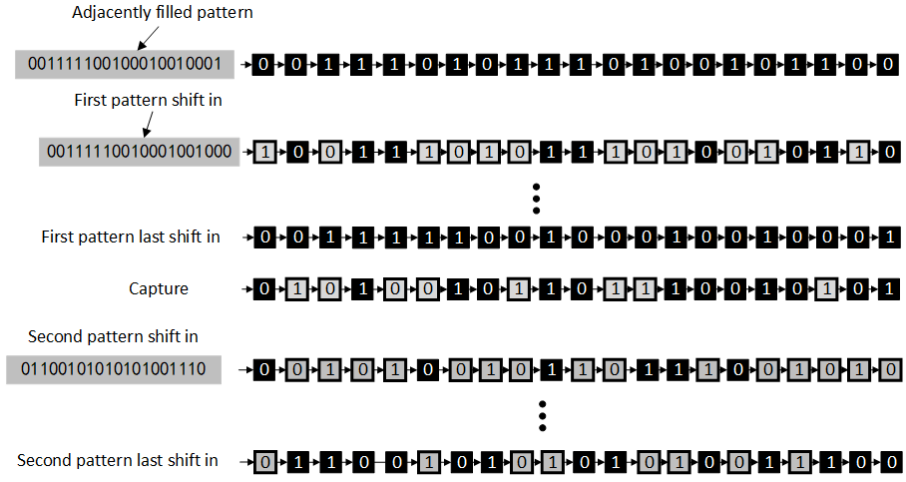

Figure 4.4: Shift in low power patterns into a regular scan chain

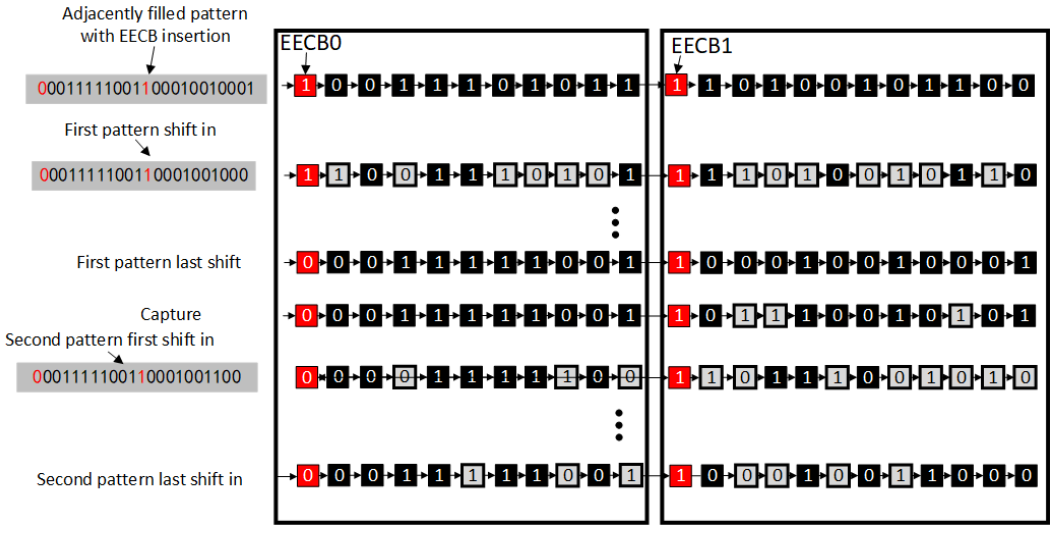

Figure 4.5: Shift in low power patterns into the segmented scan chain

first shift cycle is 13. We can also calculate the toggles for the rest of the shift cycles when the first pattern is completely shifted in, and we get a total of 263. Once the entire pattern is shifted in, and the testing process reaches the capture stage, we assume the capture response in the scan cells (SDFF0 to SDFF20) is **01010010111011100101010**. Then the total number of additional toggles in the SDFFs arising from the capture cycle is 9. Then, to shift out the data, we assume the second pattern is **0x1xx1xx010x01x011xx0**. The randomly filled pattern is **01100101010101001110 0**. The total number of toggles including the shift-in, capture, and shift-out cycles is 565.

Figure 4.4 shows the same example, but this time, a low power pattern will be shifted into a regular scan chain. Once again, assume that the scan chain has an initial value of **00110100101110101110 0**. Because the pattern that is generated from the ATPG tool is low power, assume that the pattern generated by the ATPG tool will be adjacently filled. As a result, the new pattern that is shifted into the scan chain is **10001001000010011111 00**. For every shift cycle, we count the number of toggles in the scan cells (SDFF0–SDFF20). We get 250. For the capture response, assume that we have the same capture response **01010010111011100101010** as occurred with the randomly-filled test pattern. (Obviously, this would not necessarily be true). The number of toggles during the capture cycle is 8. Note that the number of toggles during capture was approximately the same in both cases. The second pattern when adjacently filled is **01111100010001001100 0**. Shifting in this pattern while shifting out the previous capture values leads to 246 additional toggles. For shift in, capture, and shift out, we get 504, which is ((565-504)/565)*100%= 10.79% power reduction in total.

Assume the 21 bit chain is now broken into two segments; EECB0 is inserted to control SDFF0 to SDFF9; EECB1 is inserted to control SDFF10 to SDFF20. Figure 4.5 shows an example when low power patterns are shifted into a segmented scan chain controlled by EECBs. Assume SDFF0 to SDFF9 are disabled during the capture stage. As a result, the first pattern that is shifted in is **000111110011000100100 01**. The toggles during the shift and capture are 241 and 3 respectively. Note that the segment that does not capture retains the low-power shifted-in values. This will reduce toggling during the shift out of the test results. The second pattern that is shifted in is **00111110001100010011 000**, assuming

scan cells (SDFF0–SDFF10) will be disabled during capture. For shift in, capture and shift out, we get 448, which is ((565-448)/565)*100%= 20.7% power reduction in total.

We found that an ATPG tool is able to generate both stuck-at and Launch off Capture (LOC) transition fault patterns automatically for this segmented scan chain design. In the case of LOC patterns, a segment whose EECB value is 0 is disabled for both capture cycles. The ATPG tool is also able to generate patterns for a design in which an on-chip decompressor is used to fill the segmented scan chains.

4.2 EECB Insertion Procedure

To implement the proposed approach, it is necessary to split each scan chain into multiple segments that can be enabled/disabled during capture. More specifically, the following procedure in Figure 4.6 is followed for each circuit studied in this chapter.

The implementation procedure in Figure 4.6 can be illustrated with the following example and the following steps. Assume you are given a circuit with 503 scan cells, where the scan cells will be configured into 5 chains.

1. Insert balanced scan chains so that the scan chain lengths are approximately the same. In this case, there are either 100 or 101 scannable D flip-flops per chain. Assume the number of scan cells per chain are as follows. Chain 1: 101; Chain 2: 101; Chain 3: 101; Chain 4: 100; Chain 5: 100;

2. Assume the scan chains will be split evenly into two segments. (A larger number of segments is also possible.) Then the number of scan cells per segment per chain is as shown in Table 4.1.

3. Insert *EECBs* in between the scan segments by modifying the Verilog netlist according to the structure shown in Figure 4.2. As shown in Figure 4.2, one *EECB* cell and a two-input OR gate are inserted before the first scan cell of every segment. The *EECB0* shift-in signal will be the previous shift-in signal of *SDFF0*. In this case, it is the *SI* signal for that scan chain. The output of the *EECB* will fan-out to three paths: it feeds back to one of the inputs of its own mux, the shift-in signal for the first scan cell of the segment it controls, and one of the inputs into the added OR gate. The other

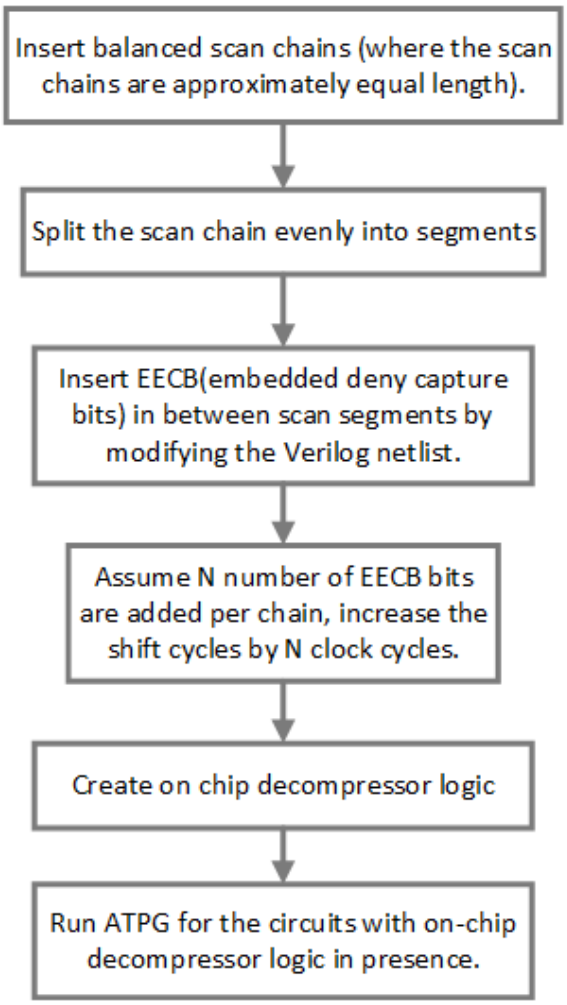

Figure 4.6: Flow chart of EECB insertion procedure.

Table 4.1: Number of scan cells per segment per chain

chain number	1	1	2	2	3	3	4	4	5	5
segment length	1	2	1	2	1	2	1	2	1	2
number of scan cells for each segment	51	50	51	50	51	50	50	50	50	50

input to the OR gate is the OR of the original *enable* signal and *SE*. In this example, we need two *EECBs* and two additional OR gates per chain, and then we can connect accordingly.

4. Change the test procedure file. In the ATPG test procedure, the minimum number of shift cycles depends on the number of scan cells in the longest functional chains and the number of *EECBs* that are being added. In this example, the original length of the scan chain is 101. Two *EECBs* are added per chain. The minimum number of shift cycles should be changed to 101+2=103.

5. Use the ATPG tool to create the on-chip decompressor with the updated test procedure and Verilog netlist.

6. Generate the on-chip decompressor test patterns automatically with the ATPG tool.

In this chapter, the power is estimated by counting the number of times each gate input or output or internal wire switches during test.

4.3 Results

To evaluate the effectiveness of our approach, data were collected for four different circuits obtained from opencores.org. The characteristics of these circuits are listed in Table 4.2.

Table 4.2: Characteristics of Benchmark Circuits

Circuit	# of scan DFFs	# of scan chains	avg. length of scan chains
des56	312	5	62-63
fm_receiver	509	5	101-102
colorconv	879	5	175-176
fpu_double	5364	5	1072-1073

For each circuit, the steps outlined in Section 4.2 were followed to generate a low power stuck-at test set and a low power transition fault test set using LOC (launch off capture).

The goal was to see how much *additional* test power reduction in both shift and capture, above and beyond that which was already obtained by the ATPG low power test pattern set, is possible with this approach. To investigate the effect that different segment lengths may have on the overall power reduction, the process was repeated for different segment lengths.

4.3.1 Stuck-At Fault Model Results

In our first set of experiments, low power test patterns that target stuck-at faults in the presence of an on-chip decompressor are applied to the circuits using the procedure in Section 4.2.

4.3.1.1 Unconstrained patterns generated with the ATPG tool

In this experiment, the ATPG tool has perfect freedom to decide whether to enable or disable a particular segment during each capture cycle. We then estimate the test power (during both shift and capture cycles) by counting the switching activities for all four circuits in Table 4.2.

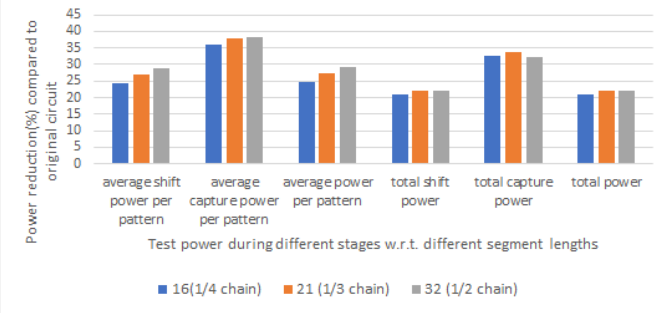

Figure 4.7: *des56* Test Power Reduction for Stuck-at Fault Patterns

Figure 4.7 shows the results for circuit *des56*. The X-axis presents different conditions under which the test power reduction is calculated:

- *Average shift power per pattern*: The sum of the switching activity across all shift cycles divided by the number of patterns.

- *Average capture power per pattern*: The sum of the switching activity across all capture cycles divided by the number of patterns.
- *Average power per pattern*: The sum of the switching activity across all shift and capture cycles divided by the number of patterns.
- *Total shift power*: The sum of the switching activity across all shift cycles for all patterns.
- *Total capture power*: The sum of the switching activity across all capture cycles for all patterns.
- *Total power*: The sum of the switching activity across all shift and capture cycles for all patterns.

The legend in the figure shows that the different colored bars represent different segment lengths. The y-axis shows the percentage of test power *reduction* compared to the original *des56* test pattern set generated for the version of the circuit without our DFT architecture. The results are presented in the same manner for the remaining three circuits in Figures 4.8, 4.9 and 4.10.

Table 4.2 shows that *fpu_double* has longer scan chains compared to the other circuits. For this reason, the experiments run on *fpu_double* and the results shown in Figure 4.10 are presented for the scan-chain partitioned into smaller fractional segments compared to the other three circuits.

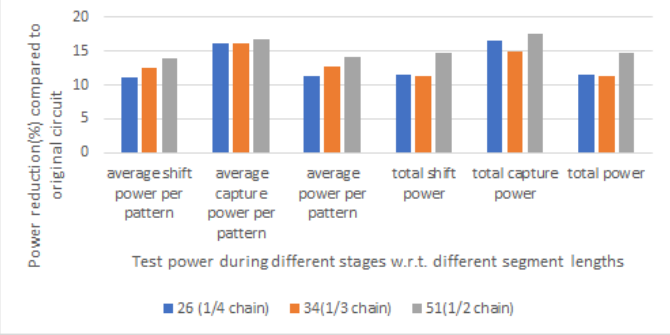

Figure 4.8: *fm_receiver* Test Power Reduction for Stuck-at Fault Patterns

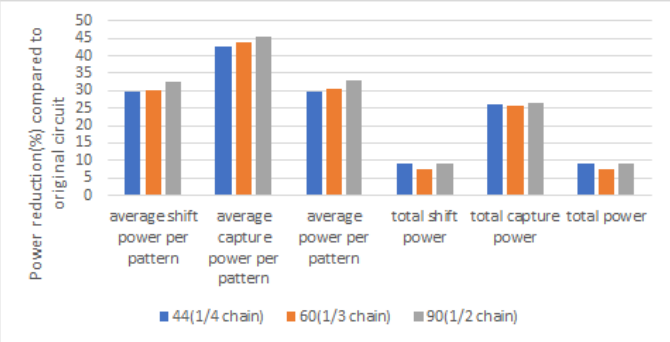

Figure 4.9: *colorconv* Test Power Reduction for Stuck-at Fault Patterns

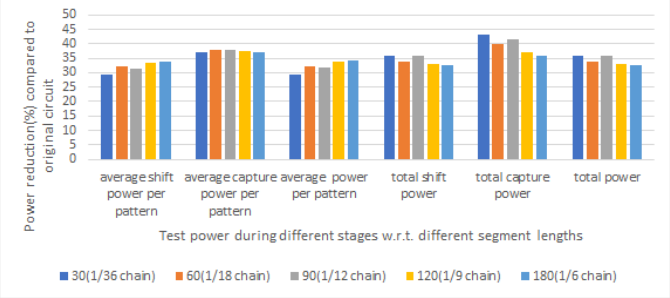

Figure 4.10: *fpu_double* Test Power Reduction for Stuck-at Fault Patterns

The results for the four circuits in Figures 4.7 – 4.10 show that the DFT architecture presented in Section 4.1 can be used to save power during test in both the shift and capture stages for the stuck-at fault model. From our results, we see that the largest of our four circuits, *fpu_double*, displays the best power reduction and can achieve up to 35% total test power reduction and almost 45% reduction for total capture power. A closer analysis of the test patterns show that *fpu_double* has the most don't care bits out of the four circuits and that a large number of patterns in the test set for *fpu_double* only detect a few hard-to-detect faults, requiring only a few segments to be enabled during capture for these patterns.

4.3.1.2 All segments are forced to capture for the first few patterns

In this experiment, we force all segments to capture during the first N patterns by constraining the values of all the EECBs to be 1 when the ATPG tool generates test patterns. The reasoning behind this involves the fact that easy-to-detect faults are more likely to be fortuitously detected in the first few patterns if all segments capture. This allows the targeting of faults later in the test set to focus on the harder-to-detect faults (when only a few segments will need to be enabled for detection). The goal is to reduce the total number of patterns needed and thus the total energy expended during test. The disadvantage is that the switching activity of the early patterns will likely be higher with this approach.

Consider the results for *des56* shown in Figure 4.11. This experiment was run with the scan chain divided into 32-bits per segment (half of the scan chain length per segment). The legend labels the bars as 0, 5, 10...60 to indicate the number of initial patterns for which all segments are forced to capture. Here, bar *0* indicates that the ATPG tool has complete control over whether a particular scan segment can capture or not; a *5* means that all scan segments are forced to capture for the first five patterns in the test pattern set. The data show that the best power reduction occurs when all segments are forced to capture for the first five patterns.

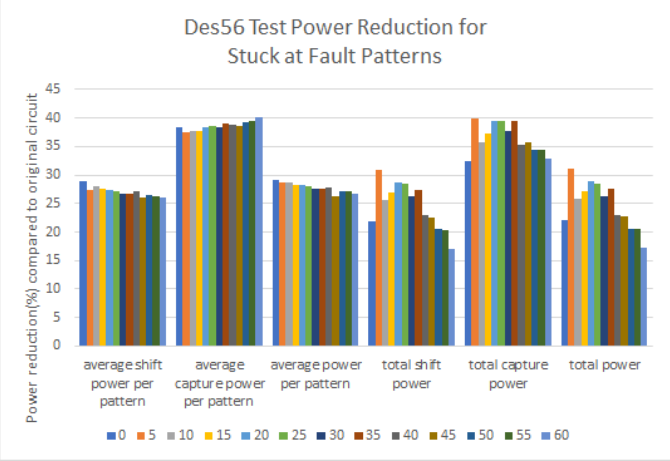

Figure 4.11: *des56* Test Power Reduction for Stuck-at Fault Patterns when all the segments are enabled during capture for the first n patterns. Each colored bar corresponds to a different value of n. (Segment length = 32 bits.)

Figure 4.12 shows the relationship between the pattern count and the total switching activity during test as the number of initial patterns (n) constrained to capture in all segments changes from 0 to 60. In each group of bars, the pattern counts or switching activities for test sets with different values of n are divided by the corresponding value for the original test set generated for the version of *des56* with no EECBs.

From Figure 4.12, it is clear that all groups of bars show a similar trend—emphasizing the importance of achieving a low pattern count test set to minimize the total energy expended during test. However, even when the test pattern count is not minimized, the proposed approach still achieves significant reduction in switching activity, as shown by the reduction that occurs across all values of n.

Note that although constraining the initial patterns to capture in all segments was useful for *des56*, it was less useful for *fm_receiver*. The majority of the faults in *fm_receiver* are in the cone of influence of only a few of the scan cells in the chains. As a result, the amount of fortuitous detection achieved by forcing all of the segments to capture in *fm_receiver* is less than for *des56*. As a result, it is necessary to consider this characteristic of the circuit when determining the appropriate value of n for test pattern generation.

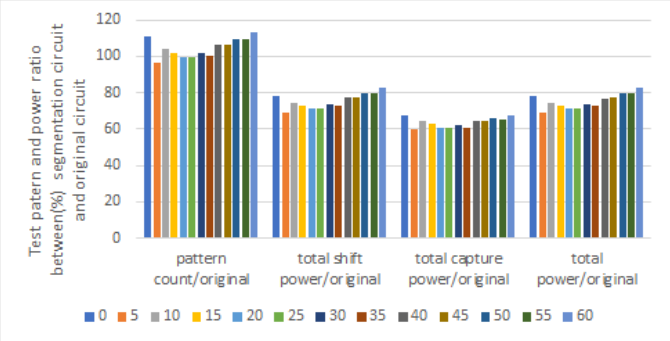

Figure 4.12: Total number of patterns and corresponding switching activity when all segments are forced to capture for the first few patterns in the pattern set for *des56*

4.3.2 Transition Fault Model Results

So far we have analyzed data collected on tests targeting stuck-at faults. In this section, we repeat the experiments and analyze power savings when targeting transition faults.

4.3.2.1 Unconstrained patterns generated with the ATPG tool

In this experiment, while targeting transition faults, the ATPG tool is given the freedom to decide whether to enable or disable a particular segment during every capture cycle. Results for the four circuits are presented in Figures 4.13–4.16. Circuits *des56, colorconv, and fpu_double* show power savings in shift and capture as well as total power. We see that our largest circuit *fpu_double* can achieve up to 37% total test power reduction.

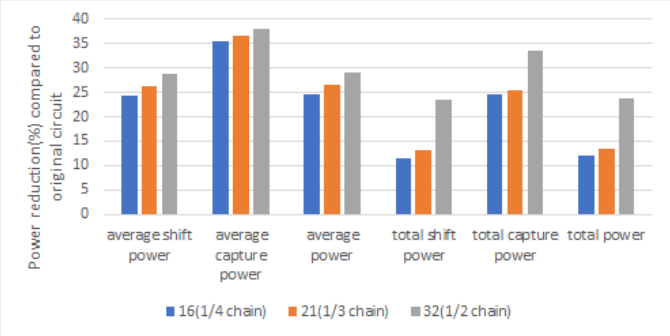

Figure 4.13: *des56* Test Power Reduction for Transition Fault Patterns

Figure 4.16 for circuit *fm_receiver* shows that on a *per-pattern basis* we see good power reduction for average shift, capture, and overall power. However, when the chains are segmented to lengths of 1/4 chain and 1/2 chain, the total shift power and total power increased slightly. This is related to a pattern count increase. Specifically, when generating a test pattern set for transition faults for *fm_receiver*, we noticed that the pattern count had increased by about 20% for the 1/4 chain and 1/2 chain scenarios compared to the original circuit without the segmented scan chain. The number of patterns for the 1/3 chain scenario was also higher but not to the same degree as the other two cases.

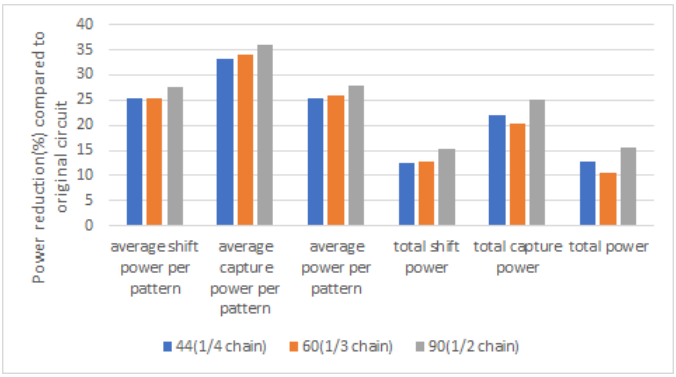

Figure 4.14: *colorconv* Test Power Reduction for Transition Fault Patterns

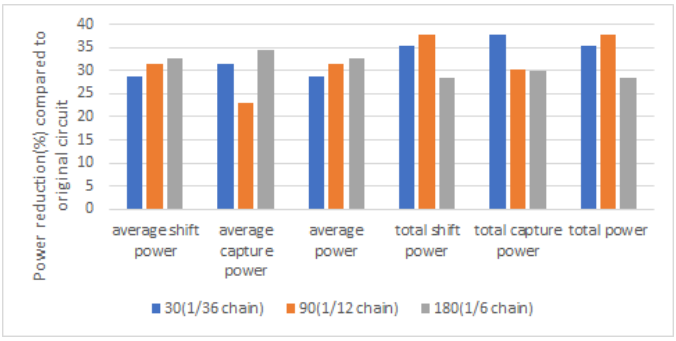

Figure 4.15: *fpu_double* Test Power Reduction for Transition Fault Patterns

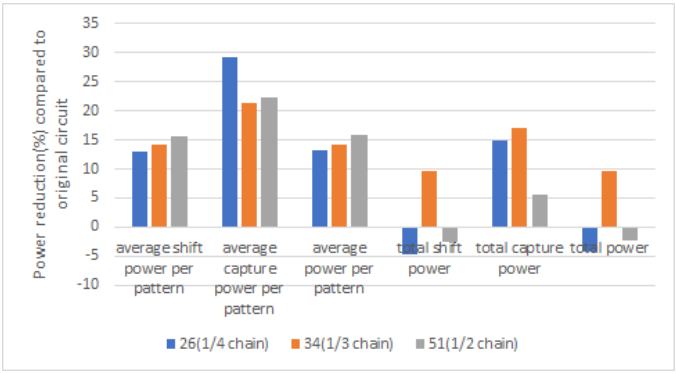

Figure 4.16: *fm_receiver* Test Power Reduction for Transition Fault Patterns

The first set of bars marked "pattern count/original" in Figure 4.17 shows this increase in pattern count over the original circuit. The other three sets of bars in the figure show the total shift power, total capture power and total power compared to the original circuit. The information in the last three groups is the same information as in Figure 4.16 but presented in a way that illustrates the direct correlation between the pattern count increase and the power savings obtained for the circuit.

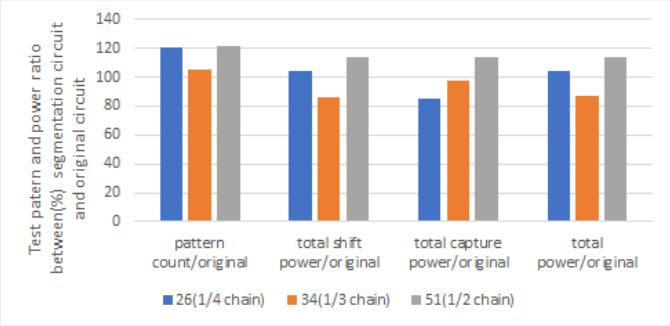

Figure 4.17: *fm_ receiver* Total test power for transition fault patterns with regard to pattern count for different amounts of scan chain segmentation

4.3.2.2 All segments are forced to capture for the first few patterns

We found that the number of transition fault test patterns generated with LOC increased when all of the segments were forced to capture for all of the values of n. As a result, while this optimization worked for stuck-at faults it doesn't appear to be useful for transition faults.

4.3.3 Overhead

4.3.3.1 Area Overhead

Different segment lengths correspond to a different area overhead. In particular, each segment requires an additional MUX-D flip-flop and an OR gate to be inserted in between the segments. In this chapter, we extract the area overhead after synthesizing and mapping

the circuits to a standard cell library. The global enable signal was included for all flops. The area overhead results are presented in Table 4.3 for *des56, fm_ receiver* and *colorconv* where the circuits were partitioned into 1/4, 1/3 or 1/2 chains, and for *fpu_ double* where the scan chains are partitioned in 1/36 chain, 1/12 chain or 1/6 chain.

Table 4.3: % Area overhead with different segment lengths

des56	L_{seg}	16(1/4 chain)	21(1/3 chain)	32(1/2 chain)
	A_o	5.2	4.65	4.13
fm_receiver	L_{seg}	26(1/4 chain)	34(1/3 chain)	51(1/2 chain)
	A_o	3.22	2.92	2.84
colorconv	L_{seg}	44(1/4 chain)	60(1/3 chain)	90(1/2 chain)
	A_o	4.05	3.69	3.49
fpu_double	L_{seg}	30(1/36 chain)	90(1/12 chain)	180(1/6 chain)
	A_o	2.44	2.31	2.24

As expected, as the segment length gets larger, the area overhead gets smaller. In Section 4.3.1 and 4.3.2, we saw that larger segment lengths (and thus lower area overhead) can achieve good power reduction. For example, the area overhead for *des56* when the segment length is 21 (1/3 of the scan chain length) is 4.65% while the power reduction for stuck-at patterns is just above 20%. As the segment length gets larger, the power reduction stays around 20% (Figure 4.7), but the area overhead drops to 4.13%. In the case of transition faults, the total power reduction doubles (Figure 4.13) when going from a segment length of 21 to 32.

In Table 4.3, the area overheads for *fpu_ double* are reported for smaller segments (1/36, 1/12, and 1/6 instead of 1/4, 1/3 and 1/2) because the average length of the scan chains (as shown in Table 4.2) for *fpu_ double* is larger than those of our remaining three circuits. We see that the area overhead for *fpu_ double* is 2.44% for segments of length 1/36 while allowing for total power savings of over 30% for both stuck-at (Figure 4.10) and transition fault (Figure 4.15) tests.

4.3.3.2 Test Time Overhead

The EECB bits inserted between the segments require extra shift cycles for each pattern. This leads to a test time overhead per pattern. In addition, if more patterns are needed once the EECBs are added, then the test cycles for the extra patterns should also be included when counting the test time overhead. The test time overhead (T_o) when scan chains have EECBs inserted is given by:

$$T_o = \frac{N_n * (CL + N_{EECB}) - CL * N_o}{(CL * N_o)} * 100\% \tag{4.1}$$

where N_{EECB} is the number of EECB bits that are inserted per scan chain, N_o is the number of patterns in the original pattern set, N_n is the number of test patterns in the new test pattern set where EECB bits are inserted and CL is the length of the longest scan chain.

The test time overhead with regard to different segment lengths for stuck-at fault patterns as well as transition fault patterns for our largest circuit *fpu_ double* is shown in Figure 4.18. For both stuck-at and transition fault test–sets, we see that the test–time overhead is under 10%. Due to the larger test pattern sets for the transition fault test generated using the LOC method compared to the stuck-at-fault tests, we see that the test overhead is a little higher. However, we also see that as the segment length increases, the test time overhead approaches 2% for stuck-at-patterns and 3% for transition patterns. (Similar results are seen for the smaller circuits.)

4.4 Conclusion

Significant switching activity reduction can be achieved by disabling capture of selected scan chain segments during test using EECBs. For our largest circuit, *fpu_ double*, using a chain length of 180 (1/6 chain), we can achieve total power savings of approximately 37% while incurring only 2.4% area overhead and only 1.9% test time overhead. We expect the approach to scale well to even larger circuits that have a high percentage of don't care values in their test pattern sets.

A significant advantage of the proposed approach is that the ATPG tool is able to generate test patterns that include the appropriate EECB values automatically without significant post processing. Furthermore, in most cases, multiple segment lengths can lead to good

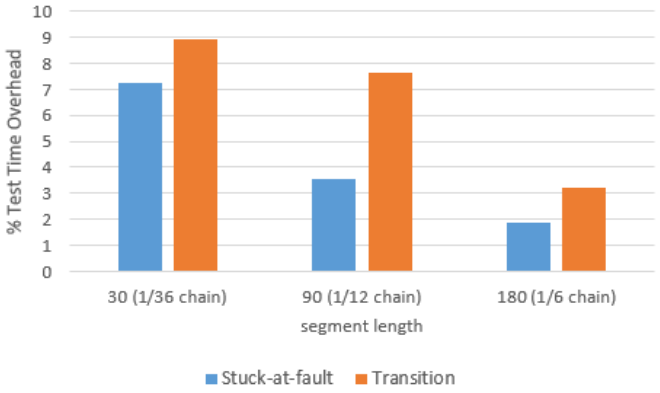

Figure 4.18: % Test time overhead for stuck-at and transition fault patterns for *fpu_ double* .

power reduction for each circuit—indicating that good trade-offs between power reduction and test time and area overhead are possible.

Chapter 5

FPGA based Tester to Enhance Field-Testing in a 3D Stacked IC

The work in this chapter was first presented in [8] and a license has been acquired to reprint this article.

With many other researchers having developed many techniques to reduce shift power and/or capture power and the two architectures we have investigated, general approaches to reduce test power have been covered. However, in certain scenarios, less work has been done, such as field test.

In field test, parts may be exposed to environments that have higher temperature than normal due to weather or other heat-producing components in other parts of the system, especially in 3D stacked ICs. Dies are stacked on top of each other within the package in 3D stacked ICs. The structure of the 3D stacked IC makes heat less likely to escape. This creates even more serious power problems during test. Using an FPGA as a field tester for a 3D stacked IC could potentially solve some of these problems.

5.1 Using an FPGA for low power testing of other dies in a 3D stack

Various methods have been developed for testing 3D stacks. For example, [50] discusses methods for scan-chain design and optimization for 3D ICs. They found that 3D scan-chain optimization achieves significant wire-length reduction compared to common 2D optimization approaches. The authors of [51] discuss DFT architecture and ATPG for interconnect test of 3D memory chips (DRAMs) and propose serial and parallel TAMs (Test Access Mechanisms) to communicate between dies. The serial TAM is used to transport test mode instructions and low-bandwidth test data, while the parallel TAM is used for high-bandwidth volume-production test data. There has also been significant research on the testing of TSVs [6], test scheduling [52], and the communication of test data between layers through the JTAG port [53].

Test approaches for chip logic in 3D stacks have generally assumed that all test data will initially be provided through the bottom die by a tester (ATE). However, 3D stacks provide other potential self-test options as well. In particular, one die in the stack may be used to test another die in the stack through the available TSV connections. The very short distances between dies in a stack can make SerDes connections very efficient. In addition, there is usually a high density of through silicon vias (TSVs) available.

If an FPGA is included in the stack for functional purposes, the high density TSVs between the FPGA and another die may not only serve as functional communication buses under normal operation, but also could have been added for performance enhancement or repair. As a result, the high bandwidth available may also allow a large number of short chains to be accessed directly for scan-based testing, reducing the overall shift cycles as well as the power dissipated during test.

The advantages of using an FPGA as a tester on a board become magnified in the 3D IC space. For example, an important issue in 3D is how and when to test each die in the stack. Bandwidth to upper dies is likely to be limited to a few pins at the base die, and the IEEE 1838 Standard [54] committee has developed protocols and methods for delivering high-bandwidth test data. These include a test access port (TAP) and TAP controller on every die, a serial boundary wrapper on every die interface to conduct interconnect testing, and a parallel port.

Bypassing traditional test and measurement equipment with FPGAs on boards has been previously shown to help significantly reduce test costs and allows high-speed testing because FPGA-based instruments can be reconfigured as needed and have direct access to the circuit [55]. FPGAs have also been embedded into SoCs (System on Chips) to provide system test capabilities [56]. Using this approach, the FPGA may be reprogrammed for different functions at different times, so the FPGA may be used to add functionality to the chip, as well as being used as an embedded tester. In the case of a 3D stacked IC, because the dies may come from different companies, dedicated embedded tester logic may be provided by each IP provider. Thus, in addition to providing a means of testing the stack, this approach may help protect the intellectual property (IP) among the different companies with IP in the stack. Furthermore, using an FPGA as a tester in a 3D stack provides significant additional

security advantages over an FPGA on a board because the inter-die connections are hidden in the stack and cannot be physically probed. As a result, test data, including test patterns, may never need to appear outside of the stack, and side channel analysis, such as power or thermal analysis, is much less likely to be effective.

Although on-chip decompressors were originally proposed to allow multiple shorter chains to be filled by only a few scan channels of input test data, when chains get too short, the number of patterns can increase—reducing or negating the improvement in test time and test energy expended [57]. Thus, a third architecture that uses an FPGA in a 3D stack as a field tester is explored in this dissertation, with the goal of reducing the total energy expended during test. In this architecture, an FPGA has been used to bypass the traditional on-chip decompressor to reduce power during test.

In our previous work [58], a tester design that was intended to take advantage of the underlying FPGA structure was introduced. Specifically, we considered the case where specific ATPG (automatic test pattern generation) patterns should be applied to the die under test and how those patterns could be efficiently stored in the lookup tables (LUTs) that form the programmable fabric of FPGAs. We explored both the FPGA resources required as well as the the amount of scan flip-flop toggling expended during scan shift. Power dissipation arising from scan shift toggling is especially important in 3D stack structures, where excess toggling may generate heat that is difficult to remove from the stack. Excessive toggling can also cause brownouts when the di/dt exceeds the capacity of power rails that have limited connections to the board. Reducing the power consumption also increases the allowable thermal budgets in a stack, allowing more ICs to pass thermal tests, increasing the number of the chips that can be stacked together, and allowing the integration of more functionality in a single stacked IC [59].

While the work presented in [58] served as a good initial exploration of the proposed approach, multiple issues remained to be explored. In this chapter we expand the work of [58] in several ways to better demonstrate the benefits of our approach. More specifically, we make the following contributions:

- We explicitly extract the interior circuit toggling during shift to better estimate dynamic power and test time using our approach.

- We investigate the trade-off between reducing power dissipation and more efficiently storing patterns in the FPGA's lookup tables by more efficiently dealing with don't cares. In particular, we consider adjacent fill merging (ADJCOM) and an "X"-retained merging algorithm (XRET) in our analysis.

- We explicitly investigate the use of multiple-input signature registers (MISRs) for capturing test responses and obtain data for MISR overhead along with the effect of aliasing on fault coverage.

5.2 Exploiting FPGA's Generic Architecture

As individual dies become more and more complex, the need for embedded instruments (such as sensors, hardware monitors, environment monitors, built-in-self test (BIST) engines, trace buffers, etc.) will only grow. There is a great possibility that they will be used not only for manufacturing test and failure or yield-analysis, but also to identify and address aging, wearout and thermal issues in the field and to verify or configure inter-die communication. An FPGA in a 3D stack may be used as a controller for these instruments or it may be used to implement some instruments, such as built-in-self-test (BIST) pattern generators.

One type of BIST pattern generator that may be implemented either in a die or on an FPGA is an LFSR-based LBIST (logic BIST) engine. Although adding weights and test points can increase the coverage of LBIST, top-off patterns may still be needed to achieve high coverage.[1] Thus, in this section, we describe one possible FPGA-based tester architecture that is capable of generating specific patterns to apply to a die-under-test (such as those that may be needed for top-off) while making use of the underlying FPGA architecture to reduce the resources needed for the design.

To meet these goals, our chosen FPGA-based tester stores the data to be shifted into the chains on different patterns into 1-bit LUTs on the FPGA. As an example, Figure 5.1 shows how the outputs of a set of LUTs are fed into a multiplexer's data inputs. The output of the multiplexer feeds into one of the scan chains on the ASIC through a TSV (possibly via a SerDes connection.) A counter is used to cycle through all of the entries in the LUTs so

[1] In order to achieve high test coverage, the number of top off patterns could be quite significant. Exploring how to decrease the top-off patterns is left for future work.

that they can be shifted out one-by-one into the chain. This same architecture is repeated for all chains in the design.

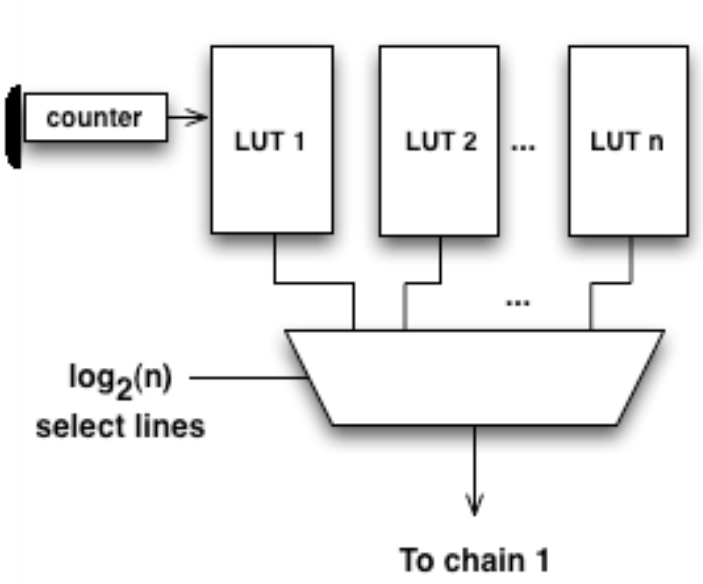

Figure 5.1: Example FPGA-based implementation for storing pattern data for a single scan chain. This is repeated for multiple chains, with LUTs possibly shared among chains.

To save on FPGA resources, we can reduce the number of LUTs by merging compatible patterns into a single LUT that can be selected multiple times. Such merging may occur both among those patterns that will eventually be fed into a single chain as well as across chains, in which case a single LUT may fanout to multiple muxes. Each scan chain would require one set of select lines for its MUX as shown in Figure 5.1.

Note that to maximize the efficiency of mapping scan data to LUTs, ideally, the scan chain length will match the number of bits available in the LUT. For example, in our experiments, we mapped our tester design to an FPGA with 5-input LUTs containing 32 bits. As a result, we used scan chains of length 32 for each of the circuits when collecting data for this architecture.

Of course, the select line data are also needed. If the length of each chain is equal to the size of a LUT, one set of select lines must be stored per mux/chain for each pattern. For longer chains, more select line values would be needed so multiple LUTs may be unloaded in sequence during scan shift. These values may be stored in the FPGA itself, in a memory located in the stack, in a memory on the board, or they may be passed to the stack by an external tester.

5.3 ADJCOM

Different approaches may be taken to merge patterns into LUTs. In particular, how Xs are filled before and/or after a merge can influence the scan shift power and the LUT overhead. Because we are interested in reducing the power during scan shift, we first look at an approach which uses adjacent fill [60] to minimize switching activity.

5.3.1 Adjacent fill merging algorithm

The LUT design process starts with a synthesized Verilog circuit netlist, which undergoes scan insertion with a scan chain size ideally equal to the size of the FPGA's lookup tables. (This will be 32 bits long in our later experiments.) An ATPG pattern set for stuck-at faults is generated in such a way that any remaining X's in the patterns are not automatically filled by the tool but are kept in the pattern set. ATPG options (such as dynamic compaction) are used to create the initial compact test set, but on-chip decompressors are not.

Table 5.1 shows an example of a pattern set divided among three chains. These ATPG generated patterns are analyzed, and any don't care "X" is filled with the value of an adjacent bit. Consider chain 1 pattern 1 (01**XX**1) shown in Table 5.1; we fill the X with 1 (the same value as the second bit) to decrease the switching activity—yielding 01**11**1. The remaining patterns are filled in a similar manner. The adjacently filled patterns are shown in Table 5.2. After adjacent fill, these patterns no longer have any don't care bits left and can now be directly assigned to LUTs.

However, direct assignment of these patterns to LUTs can be wasteful. There might be cases where, after adjacent fill, there are identical patterns within the same chain or across different chains (e.g. pattern 11111 in Table 5.2). Instead of storing 11111 multiple times we compress the data that needs to be stored in the LUTs using our Adjacent-Fill and Compress

Table 5.1: Example original pattern set

	Chain 1	Chain 2	Chain 3
Pattern 1	01XX1	100X0	XX1X1
Pattern 2	1XX11	11XX1	110XX
Pattern 3	X0XX0	1X001	1X0XX
Pattern 4	XX11X	101XX	X1XX1

Table 5.2: Example Pattern Data after Adjacent Fill

	Chain 1	Chain 2	Chain 3
Pattern 1	01111	10000	11111
Pattern 2	11111	11111	11000
Pattern 3	00000	11001	11000
Pattern 4	11111	10111	11111

(ADJCOM) algorithm illustrated in Fig 5.2.

For each chain, after we do adjacent fill, we determine what LUTs and mux connections will be needed. Specifically, after ordering the patterns in chain order, we start selecting patterns one-by-one. If the current pattern is identical to one already contained in a LUT, no new LUT needs to be allocated, and a new mux connection may be made, if required. If they are not the same, we create a new LUT, attach it to the LUT pool, and attach it to this chain's mux. In both cases, we record the appropriate select line index for this pattern so that the correct mux input (and LUT) will be selected for this pattern during test. We then check if there are any remaining patterns left. In summary, for each iteration, we need to keep track of the muxes where each LUT connects and also when that LUT should be selected (i.e., for which patterns) for each chain.

To help illustrate this ADJCOM compaction methodology, consider the following example consisting of 3 chains, 4 patterns, and 5 bits per chain, with patterns shown in Table 5.2. To reduce the LUTs and select lines required, we must merge the patterns when possible, taking the following steps:

1. Because the LUT pool is empty, we push the first pattern of Chain 1 (01111) into the

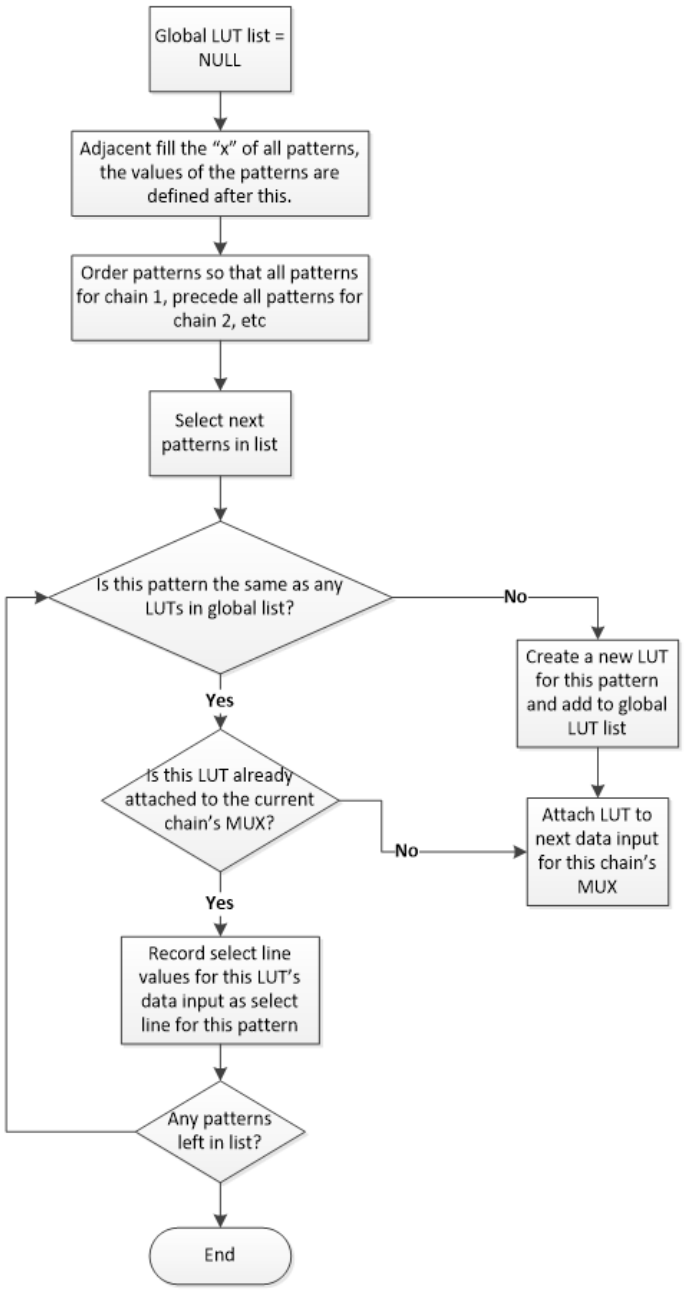

Figure 5.2: Flowchart for LUT and Select Line Reduction using ADJCOM.

LUT pool. This LUT is added to the first data input of the mux for Chain 1, and the select line value for Pattern 1, Chain 1 is set to 0.

2. Pattern 2 of Chain 1 (11111). This pattern cannot be merged with the LUT pool so we must create a new LUT. The new LUT is added to the next data input for the mux of Chain 1, and the select line value 1 for the pattern is recorded.

   Now *LUT pool: 01111, 11111. Chain 1 LUTs: 0,1; Chain 1 Select lines:0,1.*

3. Pattern 3 of Chain 1: 00000. 00000 cannot be merged with LUT0 (01111) or ADJCOM (11111). Add the pattern to the pool, attach the LUT to the third data input of the mux of Chain 1, and record the select line value.

   Now *LUT pool: 01111, 11111, 00000. Chain 1 LUTs: 0,1,2 Chain 1 Select lines:0,1,2.*

4. Pattern 4 of Chain 1: 11111. This pattern can be merged with ADJCOM (11111). Since this pattern exists in the LUT pool and is already attached to this chain's mux at data input 1, it does not need to be added to another data input. However, the select line value 1 must be recorded for this chain and pattern 4.

   Now *LUT pool: 01111, 11111, 00000. Chain 1 LUTs: 0,1,2; Chain 1 Select line values :0,1,2,1.*

5. Pattern 1 of Chain 2: 10000. This pattern cannot be merged with the existing LUT pool so we must create a new LUT. The new LUT is added to the next data input for the mux of Chain 2, and the select line value 0 for the pattern is recorded.

   Now *LUT pool: 01111, 11111, 00000, 10000. Chain 1 LUTs: 0,1,2; Chain 1 Select line values :0,1,2,1. Chain 2 LUTs: 3; Chain 2 Select line values: 0.*

This process continues until we have attempted to merge all of the patterns. The final result is shown in Figure 5.3. The merging process allows LUTs to be shared among chains and also allows the size of the muxes to be reduced when the same LUTs can be used multiple times for each chain.

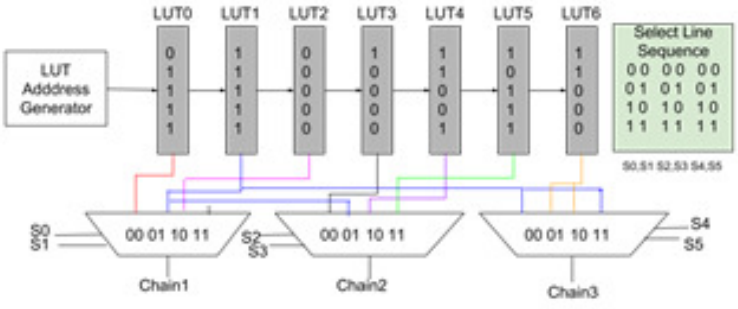

Figure 5.3: Resulting implementation for patterns shown in Table II after pattern merging.

5.3.2 FPGA Implementation Results and Analysis

To evaluate the effectiveness of our algorithm, we ran several experiments on different benchmark circuits obtained from opencores.org. These circuits were synthesized with Synopsys Design Compiler using a 90 nm ASIC library. When generating the test patterns for the circuits, both inputs and outputs of the circuits were registered.

An ATPG tool was used to insert multiple scan chains of length 32 (to match the size of the LUTs in our target FPGA) in each circuit. The scan chains contain only Primary Inputs (PIs), Primary Outputs (POs), and/or flip-flops—with the final chain possibly containing fewer scan cells when the number of flip-flops plus the PIs was not evenly divisible by 32. (The PIs and POs were registered in these experiments.) Stuck-at fault ATPG patterns were generated as well. Details regarding each of the circuits studied are provided in Table 5.3. The table lists the number of Primary Inputs (PIs), Primary outputs (POs), and Flip flops (FFs) present in the original circuit as well as the number of test patterns generated with a target test coverage at 100%, and the number of scan chains used.

Note that these circuits could very easily represent a core on a chip that needs to be tested using top-off patterns after LBIST. Furthermore, although the tester design may be used to apply top-off patterns only, in these experiments we will store and apply the entire test set for each circuit.

After a test is applied, the capture values of the flip-flops and primary outputs are shifted out to a MISR (multi-input signature register). In our experiments, we fill all the values in

Table 5.3: Characteristics of our Opencores.org Benchmark Circuits

	PIs	POs	FFs	Faults	Patterns	Chains
quad	36	25	184	7132	40	6
color	297	34	858	36534	91	29
des56	132	67	193	16050	120	14
fm	10	12	521	23408	365	18
fpu	72	70	5493	276930	254	172

the PIs and scan chains with known values, and there are no uninitialized memories or other sources of X's, so we can use the MISR as our result compactor to store the test response.

To provide a proof-of-concept implementation of our design outlined in Section 5.2, we mapped the tester architecture to a Xilinx Artix-7 (XC7A200T) FPGA device using Xilinx Vivado software. The Artix 7 series configurable logic block (CLB) provides real 6 and 5 input look-up tables (134,600 LUTs), distributed memory (2,888Kb), block RAM memory (13,140Kb), shift register (1.444Kb) logic capabilities, and fast wide multiplexers (16:1 MUX using 4 LUTs or 1 slice) for efficient FPGA fabric utilization [61].

These features of the FPGA are important for efficient implementation of our controller. Figure 5.4 shows a basic architecture of the test controller that tries to harness existing FPGA resources. The structure consists of several modules—a LUT address generator, a LUT layer, a RAM address generator, a RAM layer, a multiplexer layer, a scan register, a signature register (MISR), and a scan enable signal generator. Our controller will have a fixed set of 5-input LUTs that each store a 32-bit pattern. These LUTs will be multiplexed with wide multiplexers. To take advantage of LUT sharing as described earlier, and to reduce the total width of multiplexers as much as possible, the select lines for the multiplexers are predetermined and stored in another RAM block (implemented as either distributed RAM or block RAM on the FPGA). The test controller has three inputs (CLK, RESET, and a scan signature from the ASIC), four outputs (a scan enable signal, a reset sent to the ASIC, a registered bus feeding scan data to the ASIC via a SerDes connection, and an output that indicates whether the test passes or fails.)

As noted earlier, we ran experiments on several circuits from opencores.org to validate the effectiveness of our approach. Two separate implementations for each circuit were

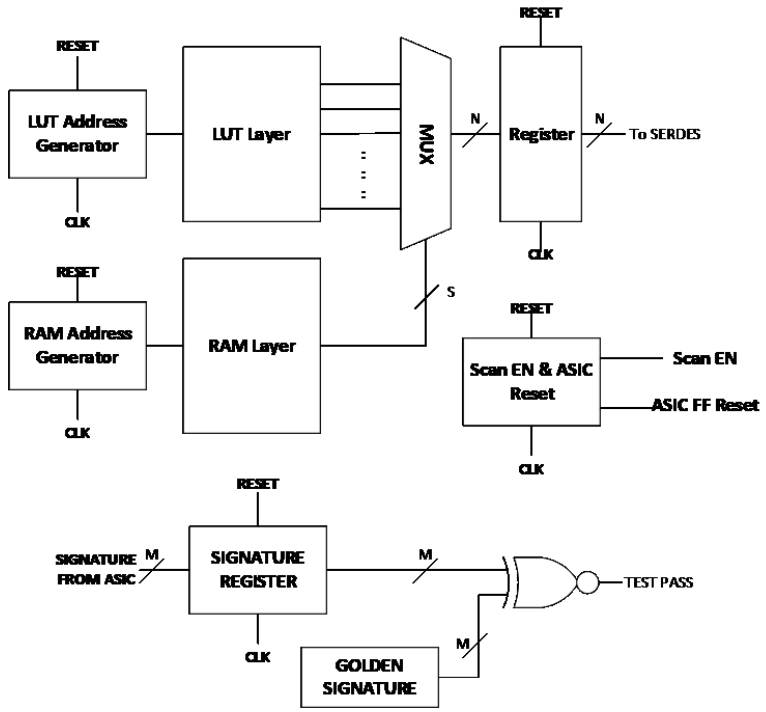

Figure 5.4: FPGA-based tester block diagram.

generated—one where all modules were implemented as distributed RAM or slice LUTs in the FPGA and a second where the mux select signals were all grouped into a larger Block RAM (BRAM) in the FPGA. For both experiments, we used Verilog HDL and synthesized it with Xilinx Vivado 17.2 with a synthesis goal set to reduce the overall system area. Results of these two experiments are shown in Tables 5.4 and 5.5. Note that the area results include all the structures shown in Figure 5.4 except the signature register, golden signature, and XNOR comparison logic. This signature logic is negligible compared to the rest of the FPGA-based tester block design.

Table 5.4 shows that the tester architecture takes up very little area on the FPGA and that the tester can be operated at a clock frequency of 164.3 to 254.5 MHz for Experiment 1. Note that the tester does not need to operate at the speed of a functional ASIC because the tester is primarily engaging in scan shift operations, which can occur at a much slower clock frequency. In fact, a slower clock frequency for scan shift is likely to be preferable to prevent thermal issues in the stack during test. The smallest circuit *quad* used negligible hardware

Table 5.4: Experiment 1: All modules are distributed RAMs/Slice LUTs

Circuits	Max Freq (MHz)	Slice LUTs	% use LUTs
quad	254.5	207	0.15%
color	210.7	2107	1.5%
des56	233.1	797	0.6%
fm	180.4	2515	1.8%
fpu	164.3	11571	9.8%

resources and was the fastest while *fpu* used the most resources (9.8% of LUTs available) and could be run at just over 164.3 MHz. In experiment 1, the speed of the circuits goes up if less LUTs are used. However, if more LUTs are used, more routing resources are required, and this may lead to longer routes and more signal delay.

Table 5.5: Experiment 2: Mux select lines implemented in block RAMs (BRAMs)

CKT	Max Freq (MHz)	Slice LUT	% use LUTs	Block RAMs	%use BRAMs
quad	212.7	160	0.1%	1	0.3%
color	220.9	974	0.7%	3	0.8%
des56	232.1	452	0.3%	1	0.3%
fm	221.5	1725	1.2%	3	0.8%
fpu	200.3	1687	1.1%	30	8.2%

Table 5.5 shows the results for Experiment 2, where we store the patterns and select lines in LUTs and BRAMs respectively. This results in fewer LUTs compared to Experiment 1 because all the LUTs of Experiment 1 that were dedicated to storing the multiplexer select line values are no longer needed. The corresponding data are now stored in one or more of the 365 available block RAMs (BRAMs) instead. Keeping the select lines in BRAMs also helps to increase the tester speed of three of the circuits. However, the small size of the *quad* circuit prevented it from taking significant advantage of the BRAMs.

5.3.3 Data Reduction using ADJCOM

Another issue we wanted to explore was how much data reduction we were able to achieve with our current FPGA-based architecture. We took a preference in reducing the number of LUTs, storing only the compressed scan chain pieces in the LUTs. As a result, the number of bits in the LUTs should be less than the original data bits. If the number of select lines needed for each chain MUX was not too large, then even storing pattern bits and select line bits should be less than the original data needed to store the test patterns.

The data obtained for our 5 circuits is shown in Table 5.6. (Note that this does not consider additional bits needed to implement the actual controller in the FPGA.) The first column corresponds to the circuit name and the second to the original amount of test data that would need to be stored. This is simply equal to:

$$32 \times \#ofchains \times \#ofpatterns \tag{5.1}$$

including padding, for chains of length 32 bits. Column 3 corresponds to the number of bits stored for pattern pieces in the LUTs and is equal to the number of LUTs identified with the algorithm in Section 5.3.1 multiplied by 32 (LUT size). Column 4 adds the data for the select line values on each pattern and is equal to the number of select lines needed for all chain muxes multiplied by the number of patterns. Column 5 corresponds to the percent reduction in data required when Columns 3 and 4 are added together and compared with the original test data shown in Column 2. We see a larger percentage reduction in the number of bits needed to store LUTs and select lines as the total original test data increases. Specifically, we see a reduction of 51% for our largest circuit. This is encouraging. Column 6 compares Column 2 and 3 to determine the percent reduction in data storage needed if only the data in the LUTs is considered. This might be significant if we are worried about the occupancy of the FPGA but are obtaining the values on the select lines from an external memory.

Finally, Column 7 compares the amount of data stored for select bits only (number of total select bits multiplied by the number of patterns) to the total number of bits in Column 2. This comparison is most appropriate from the perspective of how much data may need to be stored in an external memory for feeding to the FPGA. For four of the five circuits

tested we see this reduction in test data to be over 75%.

Table 5.6: Data Storage Reduction

CKT	Original total (bits)	LUT data (bits)	Select line (bits)	% ↓ (LUT +sel)	% ↓ (LUT only)	% ↓ (sel only)
quad	7680	6624	1260	-2.7%	14%	83%
color	84448	67424	70034	-63%	20%	17%
des56	53760	25504	9960	34%	53%	81%
fm	210240	80480	50370	38%	62%	76%
fpu	1398016	370272	308610	51%	73%	76%

We thus see that the selected FPGA-based tester architecture is highly effective at reducing the amount of test data that may need to be stored in an external memory or on the FPGA itself. Even more encouraging, the method appears to scale very well with increasing amounts of test data.

Although we were able to compact our data well enough in the previous section, the overall compaction rate is considerably less than is often achieved with the compression rate of traditional on-chip decompressors alone. Of course, it is still possible to write the decompressor's incoming channel data to LUTs or to on chip memories in the FPGA.

There are several reasons why having less compaction rate may not be a significant problem. First, as already noted, the patterns stored in the LUTs may correspond only to those top-off patterns that are needed to get coverage for random-pattern-resistant faults that are not covered by LBIST engines. This automatically reduces the test data volume that needs to be stored. Even if the number of top-off patterns required is relatively large, as we showed in Table 5.6, there are multiple approaches to storing the test data depending on the size and available resources in the FPGA and off-chip memories that can help ameliorate the issue.

In addition, one of the reasons why such decompressors are needed is to reduce the test

data bandwidth when the test inputs and outputs are limited to only a few pins. When an FPGA in a 3D stack is used, it may be possible to have many more chains on other dies accessed directly either through individual TSVs or through TSVs that are implementing SerDes. SerDes TSV channels are extremely efficient in 3D because of the very short distances between dies. This means that the test data bandwidth may automatically be higher in 3D between dies even without an on-chip decompressor, if we choose not to use one. In addition, if test patterns are going to be generated or selected within the stack so that only a subset of all potential patterns in the set are applied to better match suspected defects or operating conditions, it might be necessary to set the decompressor to bypass mode and use patterns stored in the LUTs directly instead.

Finally, thermal issues during test are likely to be very problematic in 3D because it may be more difficult for heat to escape, even with new materials proposed to enhance heat dissipation [62]. Thus, reducing switching activity during scan shift is very important. Although low power ATPG for on-chip decompressors is possible with commercial tools, some approaches to reducing scan shift toggling, such as adjacent fill, are difficult or impossible to apply in the presence of on-chip decompressors because they depend on having a large number of X's. It may be easier to get low power test patterns from our approach if enough X's remain in the patterns to perform adjacent fill.

To investigate this possibility, we collected data regarding the difference in switching activity obtained both for patterns shifted in as the output of a power-limited on-chip decompressor as well as for our original scan patterns with adjacent fill implemented after merging.

For these experiments we used scan chains of length 32 bits for all the circuits whether generating patterns with or without an on-chip decompressor. To try to make the switching activity comparison as fair as possible, more than 200 test sets were created for each circuit with different low-power parameters when patterns were generated in the presence of the on-chip decompressor. The pattern set with the lowest toggling activity that did not lead to a significant reduction in test coverage was selected for comparison against our approach. We also used low-power options to generate patterns for our approach and allowed X's to remain in the test set for merging and adjacent fill.

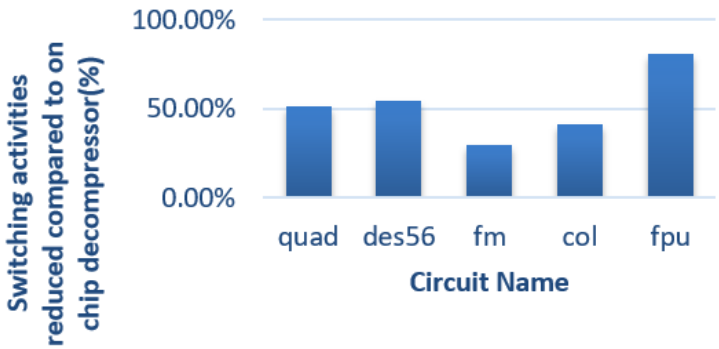

Figure 5.5: Switching activities for our method vs. on chip decompressor.

5.3.4 Switching Activities

To collect the switching activities, we apply our test set to each circuit, simulate the circuit, and extract total switching activity from every node in the circuit using VCD (value change dump) files. In each case, the switching activities of flip-flops in the chains as well as any switching activities that would have been generated in the circuit's combinational logic is also included during both shift and capture. The switching activity reduction achieved using our ADJCOM approach over the on-chip decompressor during scan test for each circuit is shown in Figure 5.5 (To make the comparison fair, the switching activities in the on-chip decompressor is not included). The switching activity reduction is computed using:

$$\%reduction = (1 - \frac{activity_{\text{ADJCOM}}}{activity_{\text{on chip decompressor}}}) \times 100\% \tag{5.2}$$

We see that each circuit shows significant reduction in total switching activity, with our largest circuit *fpu* showing an 81% reduction. One possible reason for this is that, when an on chip decompressor is used, the ATPG tool tends to create more patterns than the original pattern set.[2] Higher pattern count translates to more switching activity. Our approach requires fewer patterns. This, coupled with our use of adjacent fill, results in lower power consumption for ADJCOM. It is also encouraging that the associated tester architecture works better for the largest circuit with increasing amounts of test data: *fpu*.

[2]Note that the length of the scan chain can have a bearing on the number of patterns produced from an on chip decompressor [57] with larger chains leading to fewer patterns.

5.4 XRET

The data storage experiments of Section 5.3.3 showed that some circuits did not see as much of a benefit (e.g. circuit *color* showed a 63% increase in LUT and Select line data over the original case). In this section we propose a new algorithm called X retained merging algorithm (XRET) to efficiently deal with don't cares to improve the storage of LUTs data and select lines on an FPGA.

5.4.1 Merging with "X" in pattern set retained

In this approach, the don't care bits in the original pattern set are retained in order to achieve the maximum pattern reduction. The way the patterns are merged is different, such that it significantly improves the results regarding the number of LUTs (and therefore the area overhead and data compaction). For each chain, we analyze the patterns that will be applied to that chain and see if different patterns can be merged into a single LUT. We also look to see if patterns across different chains can be merged to reduce the total number of LUTs. Note that a pattern can only be merged with a member of the current global list of LUTs (i.e., the LUT pool), if for all bit positions of the pattern the bits are compatible between the pattern and the LUT. An X merged with a defined value (0 or 1) is replaced by the defined value in the merged LUT. In each case, we need to keep track of the muxes that each LUT connects to and when that LUT should be selected (i.e., for which patterns) for each chain.

To help illustrate this compaction methodology, consider the same patterns used in Table 5.1. To reduce the LUTs and select lines required, we must merge the patterns when possible, taking the following steps:

- Because the LUT pool is empty, we push the first pattern of Chain 1 (01XX1) into the LUT pool. This LUT is added to the first data input of Chain 1's mux, and the select line value for Pattern 1, Chain 1 is set to 0.

- Pattern 2 of Chain 1:1XX11. This pattern cannot be merged with the LUT pool so we must create a new LUT. The new LUT is added to the next data input for Chain 1's mux, and the select line value 1 for the pattern is recorded. Now *LUT pool: 01XX1, 1XX11. Chain 1's LUTs: 0,1; Chain 1's Select lines:0,1.*

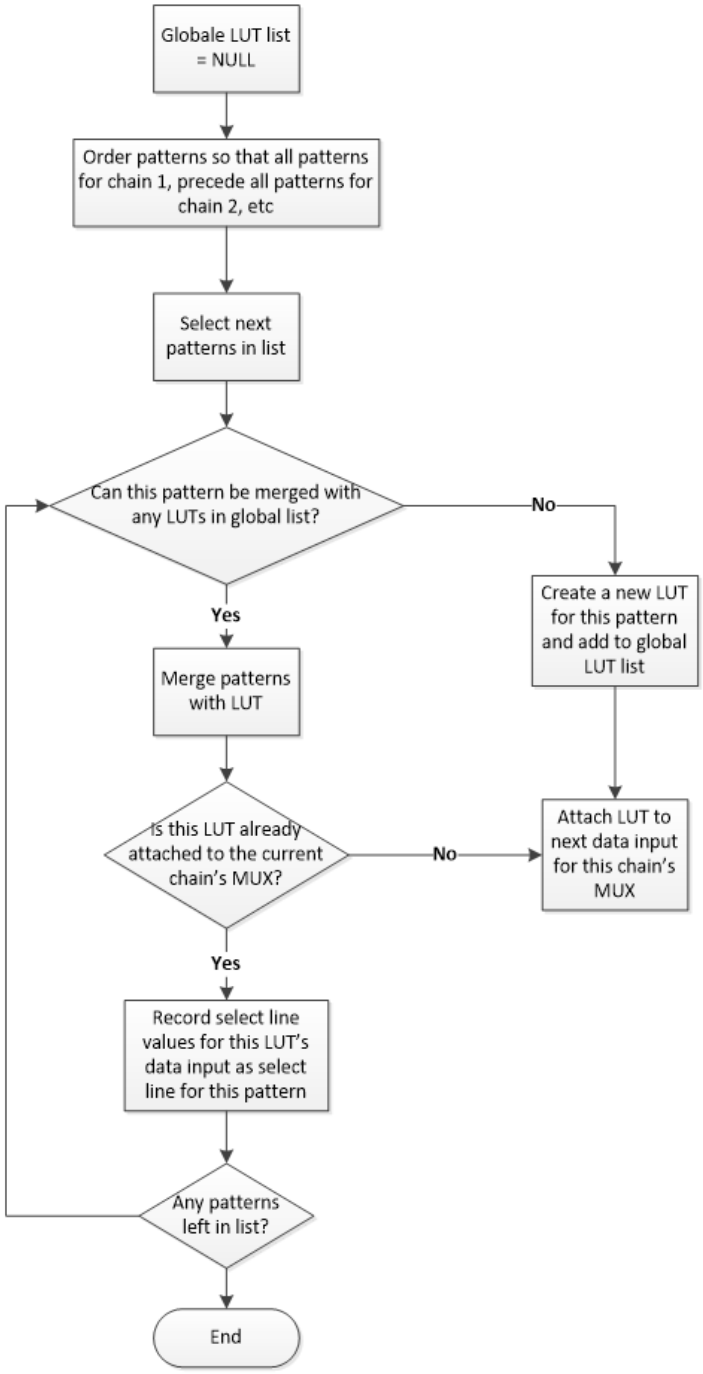

Figure 5.6: Flowchart for LUT and Select Line Reduction using XRET.

- Pattern 3 of Chain 1: X0XX0. X0XX0 cannot be merged with LUT0 (01XX1) or XRET(1XX11). Add the pattern to the pool, attach the LUT to the third data input of Chain 1's mux, and record the select line value. Now *LUT pool: 01XX1, 1XX11, X0XX0. Chain 1's LUTs: 0,1,2 Chain 1's Select lines:0,1,2.*

- Pattern 4 of Chain 1: XX11X. This pattern can be merged with LUT0 (01XX1). Create merged pattern 01111 and replace LUT0 in the pool with this merged pattern. Since LUT0 exists in the LUT pool and is already attached to this chain's mux at data input 0, it does not need to be added to another data input. However, the select line value 0 must be recorded for this chain and pattern 4. Now *LUT pool: 01111, 1XX11, X0XX0. Chain 1's LUTs: 0,1,2; Chain 1's Select line values :0,1,2,0.*

- Pattern 1 of Chain 2: 100X0. This pattern can be merged with XRET (X0XX0) to create 100X0. Replace XRET with this new merged pattern in the pool. Add XRET to Chain 2's MUX 0th data input and record 0 as the select line value for Chain 2, pattern 1.

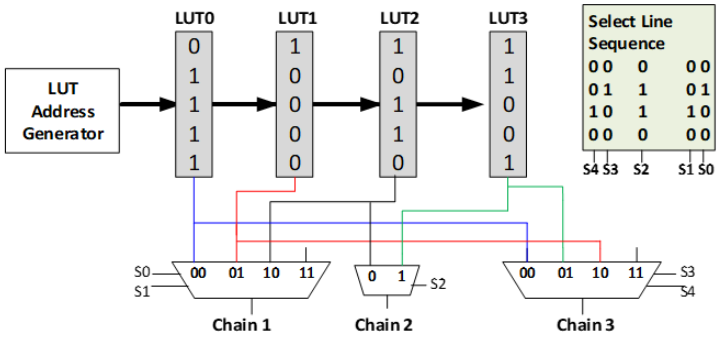

Figure 5.7: Resulting implementation for patterns shown in Table I after pattern merging.

This process continues until we have attempted to merge all of the patterns. To store the final data into the LUTs, we replace any remaining don't cares with 1s and 0s using the

adjacent fill technique.

This gives us our final LUT pool: 01111, 10000, 10110, 11001. The resulting implementation is illustrated in Figure 5.7. For this example, we see both a reduction in the LUT bits as well as the select line bits compared to the implementation after ADJCOM (Figure 5.3).

Table 5.7 shows that this new merging algorithm takes up less area on the FPGA compared to the adjacent fill merging algorithm and that the tester can be operated at a clock frequency of 167.6 to 270.2 MHz for Experiment 1 where all modules were stored in distributed RAMs or slice LUTs. Again, the tester does not need to operate at the speed of a functional ASIC. The circuits used negligible hardware resources. Our largest circuit *fpu* showed the most reduction in LUT use (down from 11571 in Table 5.4) to 2996). Other conclusions that were drawn from Table 5.4 can also be applied here.

We also see a similar pattern to that in Table 5.5 when we run the XRET algorithm and store the Mux select lines in block RAMS instead of distributed RAMs. The results are shown in Table 5.8. We see across all five circuits a slight improvement in the max clock frequency and a slight increase in the slice LUT count compared to ADJCOM where the Xes in the ATPG patterns are filled adjacently first before merging.

Table 5.7: Experiment 1—All modules are distributed RAMs/Slice LUTs

CKT	Max Freq (MHz)	Slice LUTs	% use LUTs
quad	270.2	175	0.12%
color	232.4	1182	0.8%
des56	240.3	578	0.4%
fm	168.2	2074	1.5%
fpu	167.6	2996	2.4%

5.4.2 Data Reduction using XRET

Using this new XRET algorithm, we repeat the data storage experiments described earlier in Section 5.3.3 for the benchmark circuits. The data storage reduction results for the circuits

Table 5.8: Experiment 2— Mux select lines implemented in block RAMs (BRAMs)

CKT	Max Freq (MHz)	Slice LUT	% use LUTs	Block RAMs	%use BRAMs
quad	229.1	124	0.08%	1	0.3%
color	251.4	847	0.6%	3	0.8%
des56	273.1	397	0.6%	1	0.3%
fm	247.8	796	0.6%	3	0.8%
fpu	223.6	1387	1.2%	30	8.2%

are shown in Table 5.9.

Table 5.9: Data Storage Reduction

CKT	original total (bits)	LUT data (bits)	Select line (bits)	% ↓ (LUT +sel)	% ↓ (LUT only)	% ↓ (sel only)
quad	7680	5600	1224	11%	27%	84%
color	84448	37824	68951	-26%	55%	18%
des56	53760	18496	9480	48%	66%	82%
fm	210240	66368	49275	45%	68%	77%
fpu	1398016	44384	284988	76%	96.8%	79.6%

Comparing the results seen in Table 5.9 with the ones obtained using ADJCOM (Table 5.6), we can see that XRET has a much higher compaction rate when the "Xs" are retained during the merging for all of our circuits across all three storage scenarios (LUT+select line bits, LUT only, and select lines only).

5.4.3 Switching activities

The switching activities reduction during scan test for each circuit is shown in Figure 5.8. The blue bars (ADJCOM) and red bars (XRET) show the percentage of power reduction compared to an on-chip decompressor. Both approaches achieve a good amount power reduction. We see that ADJCOM achieved a slightly higher power reduction compared to

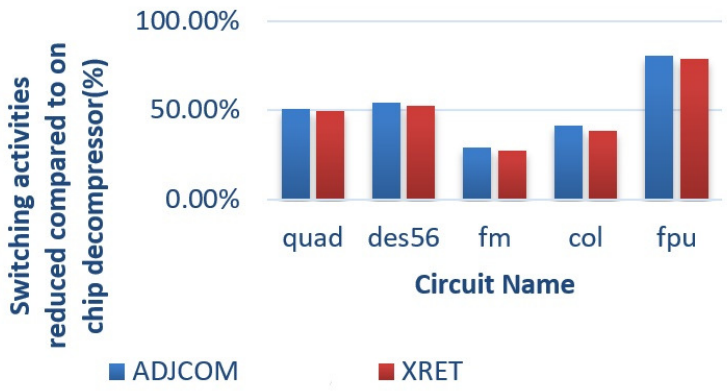

Figure 5.8: Switching Activities for our methods vs. on-chip decompressor

XRET. However, the XRET approach can still achieve almost the same amount of power reduction as ADJCOM and while achieving a much better compaction rate. As a result, XRET is a very good option when the tester needs to achieve good compaction rates while still prioritizing thermal issues during test. Furthermore, we consider XRET our best option unless power reduction is the main concern and there are plenty of FPGA resources available.

In order to obtain a more direct visualization of the power dissipation, we used Cadence Encounter Test to automatically generate the layout for all five circuits and mapped the switching to the location where the cell is located. Figure 5.9 shows an IC floorplan with total switching activity during test for *fpu*. Note that the two 2D subplots are divided into 100×100 squares, where each square could have one or even hundreds of cells. Red squares correspond to areas of high switching activity while purple/blue squares correspond to low switching activity areas. The IC floorplan shows that the switching activity is not only low in the case of XRET but areas of red spots are almost non-existent compared to the on-chip decompressor case.

Note that the switching activity corresponds to the total switching activity in each block throughout test in both cases. The floor plans are identical (i.e. the mapping of logic to the floorplan is the same in both the left and the right square), and the on-chip decompressor itself is not included in the floorplan on the right. This was done to make the comparison more fair. If the on-chip decompressor were included in the floorplan, even more total

switching activity would be seen in the circuit on the right.

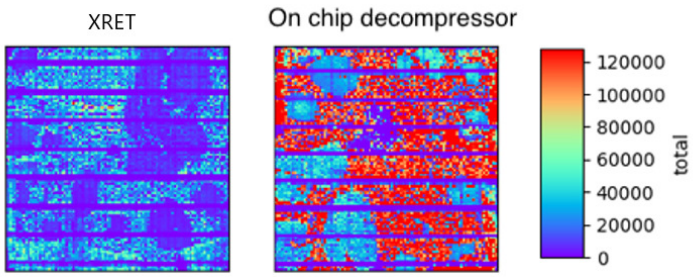

Figure 5.9: Switching activities in IC floorplan for *fpu* using XRET versus an on-chip decompressor. The legend shows switching activity scale – red corresponds to high switching and purple/blue low switching activity.

5.5 Test Response and Test Time

The test architecture shown in Figure 5.4 contains the LUTs as well as the select lines for the muxes generated using either the ADJCOM or XRET algorithms. These are used to apply the compacted test patterns to the scan chains. The test response is then captured as a signature using a multiple-input signature register (MISR). The use of a MISR is common in DFT architectures to compress the test responses. Their primary downside arises when unknown values may be present in the test responses due to partial scan designs, memory elements, etc., that lead to unknown values. In our case, no unknowns will be present in any of our benchmark circuits; however, if they were present in a design, previously proposed approaches, such as X-compact [63] could be used to help prevent unknown values from propagating into and corrupting the signature.

Unfortunately, MISRs are also known to cause some loss of coverage due to aliasing [64]. In this section, we investigate the use of a MISR and its effect on the test coverage for the benchmark circuits and patterns studied. We will show that the coverage loss is reasonable and is in line with normal coverage loss ($\leq$5%) due to a MISR [65].

Table 5.10: Test coverage before and after using MISR

Circuit	Test coverage w/o MISR	Test coverage with MISR	MISR Length	MISR XNOR Tap Points
quad	99.57%	98.21%	11	9,11
color	100.00%	96.80%	34	1,2,27,34
des56	99.98%	98.87%	19	1,2,6,19
fm	99.93%	99.77%	23	18,23
fpu	99.18%	97.82%	177	172,174,175,177

For our experiments, one MISR per circuit is being used. The MISR length is dependent on the number of chains in the circuit and is computed as:

$$\text{MISR Length} = \#\text{of chains} + 5. \tag{5.3}$$

For example, consider circuit *quad*. Table 5.3 earlier showed that it had 6 scan chains; the number of bits in the MISR is 11 for this circuit. Each MISR is created using taps for XNOR gates using a characteristic polynomial and tap points based on [66]. Table 5.10 reports the size of the MISR as well as the test coverage achieved before and after using the MISR. We see an average coverage reduction of only 1.4% across the five circuits with coverage reduction ranging from 0.16% for *fm* to 3.2% for *color*.

Our method also uses less test time for the benchmark circuits than using an on-chip decompressor for 32-bit scan chains. Test time is especially important in field testing because a device must be taken offline to perform the test. Because the FPGA programming to implement the tester can be done without the circuit-under-test being taken offline, we do not include the time required to program the FPGA in our analysis. Figure 5.10 shows the percentage of the test time used by the on-chip decompressor that is needed by our FPGA–based approach. Compared to an on chip decompressor for the same scan chain length, we can reduce the test time by 38% to 90%.

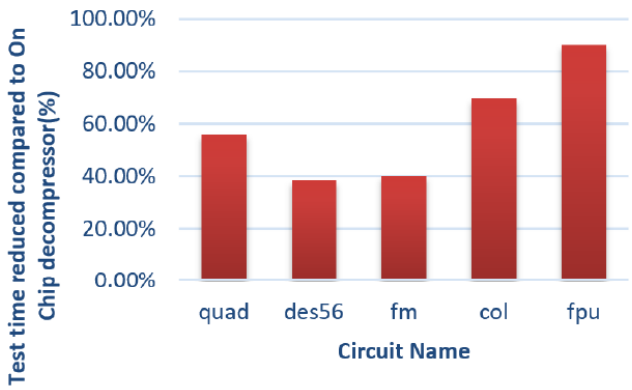

Figure 5.10: Test time reduction:our Method vs. on chip Decompressor.

5.5.1 Conclusions

In this chapter, we have explored some of the advantages of using an existing FPGA as a tester in a 3D stack. We have implemented two different merging algorithms (ADJCOM and XRET) for an FPGA-based tester design. The two methods require a very small fraction of FPGA resources for the circuits studied. We also see a reduction in the switching activity as well as test time when compared to on-chip decompressors for both methods.

Furthermore, the investigated technique can take advantage of the high TSV bandwidth that is likely possible in 3D die stacks to transmit data to multiple chains in parallel. In general, most of these advantages should also carry over into the 2.5D space.

Chapter 6
Conclusions

Due to the large amount of previous work focused on reducing shift power during test, we primarily focused on reducing capture power. In our work on the first architecture, we have shown that very high toggling reduction can be achieved during capture cycles even when we start with patterns that have been created by an ATPG tool to achieve low power. Note that because we are disabling capture in scan chain segments, any effort made by the ATPG algorithm to reduce shift power for those segments for the pattern shifted into the chain will perform a "double duty". A reduced number of transitions during scan in will lead to a reduced number of transitions during scan out for the bits in those disabled segments because the values will be identical during shift in and shift out. However, because of limitations on the number of pins available for test input and output in ASICs, the extra pin needed for the control register becomes quite expensive. As a result, we explored another related architecture that removed this limitation.

In the second architecture, we removed the extra scan input pins needed for the control bits by embedding the control bits within the existing functional scan chains. Another significant advantage for this approach is that the ATPG tool can automatically generate the required values for those control bits and remove the large amount of post processing time that is required when implementing the first architecture. We also showed that we can reduce both shift and capture power during test in the presence of an on-chip decompressor and have extended our analysis to the testing of transition faults.

In the last architecture, we have explored some of the advantages of using an existing FPGA as a field tester in a 3D stacked IC because excessive heat buildup is especially problematic in 3D stacks. In this work, we made good use of the existing FPGA fabric when implementing the tester. In addition, the investigated technique can take advantage of the high TSV bandwidth that is likely possible in 3D die stacks to transmit data to multiple

chains in parallel. This allowed us to implement very short chains with a corresponding savings in test time and total energy expended during test.

In the optimizations described in this dissertation, we have implemented two different merging algorithms (ADJCOM and XRET). The two methods require a very small fraction of FPGA resources for the circuits studied. We also see a reduction in the switching activity as well as test time when compared to on-chip decompressors for both methods. In general, many of these advantages should also carry over into the 2.5D space.

BIBLIOGRAPHY

[1] N. Nicolici, "Power minimisation techniques for testing low power vlsi circuits," Ph.D. dissertation, 2000, copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Analyte descriptor - Bibliographic data provided by EThOS, the British Library's UK thesis service: https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.561421; Last updated - 2021-05-20. [Online]. Available: http://proxy.libraries.smu.edu/login?url=https://www-proquest-com.proxy.libraries.smu.edu/dissertations-theses/power-minimisation-techniques-testing-low-vlsi/docview/301610134/se-2?accountid=6667 x, 8

[2] E. khayat Moghaddam, "On low power test and low power compression techniques," Ph.D. dissertation, 2011. [Online]. Available: http://proxy.libraries.smu.edu/login?url=https://www-proquest-com.proxy.libraries.smu.edu/dissertations-theses/on-low-power-test-compression-techniques/docview/882298536/se-2?accountid=6667 x, 8, 13, 14, 16, 19, 20

[3] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, Kun-Han Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide, and Jun Qian, "Embedded deterministic test for low cost manufacturing test," in *Proc. International Test Conference (ITC)*, Oct 2002, pp. 301–310. x, 5, 6, 15, 16, 17

[4] K. M. Butler, J. Saxena, A. Jain, T. Fryars, J. Lewis, and G. Hetherington, "Minimizing power consumption in scan testing: pattern generation and dft techniques," in *2004 International Conference on Test*, Oct 2004, pp. 355–364. 3, 21

[5] A. Grochowski, D. Bhattacharya, T. Viswanathan, and K. Laker, "Integrated circuit testing for quality assurance in manufacturing: history, current status, and future trends," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 44, no. 8, pp. 610–633, 1997. 8

[6] C. Wang, J. Zhou, R. Weerasekera, B. Zhao, X. Liu, P. Royannez, and M. Je, "Bist methodology, architecture and circuits for pre-bond tsv testing in 3d stacking ic systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 1, pp. 139–148, Jan 2015. 13, 57

[7] C. Krishna, A. Jas, and N. Touba, "Test vector encoding using partial lfsr reseeding," in *Proceedings International Test Conference 2001 (Cat. No.01CH37260)*, 2001, pp. 885–893. 13

[8] Y. Sun, F. Zhang, H. Jiang, K. Nepal, J. Dworak, T. Manikas, and R. Bahar, "Re-purposing fpgas for tester design to enhance field-testing in a 3d stack," *Journal of Electronic Testing*, vol. 39, Dec. 2011. 14, 57

[9] C. Claus, R. Ahmed, F. Altenried, and W. Stechele, "Towards rapid dynamic partial reconfiguration in video-based driver assistance systems," in *Reconfigurable Computing: Architectures, Tools and Applications*, P. Sirisuk, F. Morgan, T. El-Ghazawi, and H. Amano, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 55–67. 14

[10] E. Sperling, "Thinking outside the chip," *Semiconductor Engineering*, 2019, https://semiengineering.com/thinking-outside-the-chip/ Accessed: 2019-11-26. 15

[11] J. Xie and D. Patterson, "Realizing 3D IC integration with face-to-face stacking," *Chip Scale Review*, vol. 17, no. 3, pp. 16–19, Oct 2013. 15

[12] Xilinx, "3D ICs," https://www.xilinx.com/products/silicon-devices/3dic.html, accessed: 2019-07-27. 15

[13] P. Dorsey, "White paper: Xilinx stacked silicon interconnect technology delivers breakthrough fpga capacity, bandwidth, and power efficiency," Xilinx, Tech. Rep., 2010. 15

[14] A. L. Crouch, J. C. Potter, A. Khoche, and J. Dworak, "Fpga-based embedded tester with a p1687 command, control, and observe-system," *IEEE Design Test*, vol. 30, no. 5, pp. 6–14, Oct 2013. 15

[15] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Testing for Single Stuck Faults*, 1990, pp. 181–287. 17

[16] J. A. Waicukauski, E. Lindbloom, B. K. Rosen, and V. S. Iyengar, "Transition fault simulation," *IEEE Design Test of Computers*, vol. 4, no. 2, pp. 32–38, 1987. 18

[17] J. Savir, "Skewed-load transition test: Part i, calculus," in *Proceedings International Test Conference 1992*. Los Alamitos, CA, USA: IEEE Computer Society, sep 1992, p. 705. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TEST.1992.527892 18, 19

[18] J. Savir and S. Patil, "Broad-side delay test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 8, pp. 1057–1064, 1994. 18, 19

[19] N. Li, E. Dubrova, and G. Carlsson, "A scan partitioning algorithm for reducing capture power of delay-fault lbist," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, ser. DATE '15. San Jose, CA, USA: EDA Consortium, 2015, pp. 842–847. [Online]. Available: http://dl.acm.org/citation.cfm?id=2755753.2755944 20

[20] A. Bosio, P. Girard, and A. Virazel, "Test of low power circuits: Issues and industrial practices," in *2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Dec 2016, pp. 524–527. 20

[21] N. Nicolici and X. Wen, "Embedded tutorial on low power test," in *12th IEEE European Test Symposium (ETS'07)*, 2007, pp. 202–210. 20

[22] M. Pedram, "Power minimization in ic design: Principles and applications," vol. 1, no. 1, 1996. 20

[23] Y. Higami, S. Kobayashi, and Y. Takamatsu, "A method to reduce power dissipation during test for sequential circuits," in *Proceedings of the 11th Asian Test Symposium, 2002. (ATS '02).*, 2002, pp. 326–331. 20

[24] J. T. Tudu, E. Larsson, V. Singh, and V. D. Agrawal, "On minimization of peak power for scan circuit during test," in *2009 14th IEEE European Test Symposium*, May 2009, pp. 25–30. 21

[25] S. Bhunia, H. Mahmoodi, D. Ghosh, S. Mukhopadhyay, and K. Roy, "Low-power scan design using first-level supply gating," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 3, pp. 384–395, March 2005. 21

[26] S. Gerstendorfer and H. . Wunderlich, "Minimized power consumption for scan-based bist," in *International Test Conference 1999. Proceedings (IEEE Cat. No.99CH37034)*, Sep. 1999, pp. 77–84. 21

[27] X. Zhang and K. Roy, "Power reduction in test-per-scan bist," in *Proceedings 6th IEEE International On-Line Testing Workshop (Cat. No.PR00646)*, July 2000, pp. 133–138. 21

[28] A. S. Abu-Issa, "Energy-efficient scheme for multiple scan-chains bist using weight-based segmentation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 3, pp. 361–365, March 2018. 21

[29] M. Filipek, Y. Fukui, H. Iwata, G. Mrugalski, J. Rajski, M. Takakura, and J. Tyszer, "Low power decompressor and prpg with constant value broadcast," in *2011 Asian Test Symposium*, Nov 2011, pp. 84–89. 21

[30] D. Czysz, M. Kassab, X. Lin, G. Mrugalski, J. Rajski, and J. Tyszer, "Low-power scan operation in test compression environment," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 11, pp. 1742–1755, Nov 2009. 21

[31] X. Wen, Y. Yamashita, S. Morishima, S. Kajihara, , K. K. Saluja, and K. Kinoshita, "Low-capture-power test generation for scan-based at-speed testing," in *IEEE International Conference on Test, 2005.*, Nov 2005, pp. 10 pp.–1028. 21

[32] Y. Bonhomme, P. Girard, L. Guiller, C. Landrault, and S. Pravossoudovitch, "A gated clock scheme for low power scan testing of logic ics or embedded cores," in *Proceedings 10th Asian Test Symposium*, Nov 2001, pp. 253–258. 21

[33] R. Sankaralingam, B. Pouya, and N. A. Touba, "Reducing power dissipation during test using scan chain disable," in *Proceedings 19th IEEE VLSI Test Symposium. VTS 2001*, April 2001, pp. 319–324. 21

[34] D. Czysz, G. Mrugalski, N. Mukherjee, J. Rajski, P. Szczerbicki, and J. Tyszer, "Deterministic clustering of incompatible test cubes for higher power-aware edt compression," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 8, pp. 1225–1238, Aug 2011. 21

[35] X. Wen, Y. Yamashita, S. Morishima, S. Kajihara, , K. K. Saluja, and K. Kinoshita, "Low-capture-power test generation for scan-based at-speed testing," in *IEEE International Conference on Test, 2005.*, Nov 2005, pp. 10 pp.–1028. 21

[36] E. K. Moghaddam, J. Rajski, S. M. Reddy, X. Lin, N. Mukherjee, and M. Kassab, "Low capture power at-speed test in edt environment," in *2010 IEEE International Test Conference*, Nov 2010, pp. 1–10. 21

[37] and S. M. Reddy and I. Pomeranz, "On generating pseudo-functional delay fault tests for scan designs," in *20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'05)*, Oct 2005, pp. 398–405. 21

[38] Y. Chung and J. Rau, "Low-capture-power x-filling method based on architecture using selection expansion," in *2018 IEEE International Conference on Applied System Invention (ICASI)*, April 2018, pp. 102–104. 21

[39] W. Wang, J. Kuang, and Z. You, "Achieving low capture and shift power in linear decompressor-based test compression environment," *Microelectronics Journal*, vol. 43, pp. 134–140, Dec. 2012. 21

[40] G. Mrugalski, N. Mukherjee, J. Rajski, D. Czysz, and J. Tyszer, "Compression based on deterministic vector clustering of incompatible test cubes," in *2009 International Test Conference*, Nov 2009, pp. 1–10. 21

[41] R. Shaikh, P. Wilson, K. Agarwal, H. V. Sanjay, R. Tiwari, K. Lath, and S. Ravi, "At-speed capture power reduction using layout-aware granular clock gate enable controls," in *2014 International Test Conference*, Oct 2014, pp. 1–10. 22

[42] L. Lee, C. He, and W. Tseng, "Deterministic atpg for low capture power testing," in *2012 13th International Workshop on Microprocessor Test and Verification (MTV)*, Dec 2012, pp. 24–29. 22

[43] Z. You, J. Huang, M. Inoue, J. Kuang, and H. Fujiwara, "Capture in turn scan for reduction of test data volume, test application time and test power," in *2010 19th IEEE Asian Test Symposium*, Dec 2010, pp. 371–374. 22

[44] D. Xiang, X. Wen, and L. Wang, "Low-power scan-based built-in self-test based on weighted pseudorandom test pattern generation and reseeding," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 3, pp. 942–953, March 2017. 22

[45] L. Whetsel, "Adapting scan architectures for low power operation," in *Proceedings International Test Conference 2000 (IEEE Cat. No.00CH37159)*, Oct 2000, pp. 863–872. 22

[46] Z. You, T. Iwagaki, M. Inoue, and H. Fujiwara, "A low power deterministic test using scan chain disable technique," *IEICE Transactions on Information and Systems*, vol. E89D, 06 2006. 22

[47] E. Arvaniti and Y. Tsiatouhas, "Low-power scan testing: A scan chain partitioning and scan hold based technique," *Journal of Electronic Testing*, vol. 30, no. 3, pp. 329–341, Jun 2014. [Online]. Available: https://doi.org/10.1007/s10836-014-5453-9 22

[48] © 2019 IEEE. Reprinted, with permission, from [Y. Sun and H.Jiang and L. Ramakrishnan and M.Segal and K. Nepal and J. Dworak and T. Manikas and I. Bahar, "Test Architecture for Fine Grained Capture Power Reduction," 2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2019, pp. 558-561, doi: 10.1109/ICECS46596.2019.8964790.]. 23

[49] Y. Sun, H. Jiang, L. Ramakrishnan, J. Dworak, K. Nepal, T. Manikas, and R. Iris Bahar, "Low power shift and capture through atpg-configured embedded enable capture bits," in *Accepted for publication in Proc. IEEE International Test Conference (ITC)*, 2021, pp. 1–5. 38

[50] X. Wu, P. Falkenstern, K. Chakrabarty, and Y. Xie, "Scan-chain design and optimization for three-dimensional integrated circuits," *J. Emerg. Technol. Comput. Syst.*, vol. 5, no. 2, pp. 9:1–9:26, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1543438.1543442 57

[51] S. Deutsch, B. Keller, V. Chickermane, S. Mukherjee, N. Sood, S. K. Goel, J. Chen, A. Mehta, F. Lee, and E. J. Marinissen, "DfT architecture and ATPG for interconnect tests of JEDEC wide-I/O memory-on-logic die stacks," in *Proc. IEEE International Test Conference (ITC)*, Nov 2012, pp. 1–10. 57

[52] S. K. Roy, P. Ghosh, H. Rahaman, and C. Giri, "Session based core test scheduling for 3D SOCs," in *Proc. IEEE Computer Society Annual Symposium on VLSI*, July 2014, pp. 196–201. 57

[53] Y. Fkih, P. Vivet, B. Rouzeyre, M. Flottes, and G. Di Natale, "A JTAG based 3D DfT architecture using automatic die detection," in *Proc. 9th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, June 2013, pp. 341–344. 57

[54] "Ieee standard for test access architecture for three-dimensional stacked integrated circuits," *IEEE Std 1838-2019*, pp. 1–73, 2020. 58

[55] I. Aleksejev, S. Devadze, A. Jutman, and K. Shibin, "Virtual reconfigurable scan-chains on fpgas for optimized board test," in *Proc. 16th Latin-American Test Symposium (LATS)*, March 2015, pp. 1–6. 58

[56] S. Devadze, A. Jutman, I. Aleksejev, and R. Ubar, "Fast extended test access via JTAG and FPGAs," in *Proc. International Test Conference*, Nov 2009, pp. 1–7. 58

[57] K. Chakravadhanula, V. Chickermane, P. Cunningham, B. Foutz, D. Meehl, L. Milano, C. Papameletis, D. Scott, and S. Wilcox, "Advancing test compression to the physical dimension," in *Proc. IEEE International Test Conference (ITC)*, Oct 2017, pp. 1–10. 59, 73

[58] F. Zhang, Y. Sun, X. Shen, K. Nepal, J. Dworak, T. Manikas, P. Gui, R. I. Bahar, A. Crouch, and J. Potter, "Using existing reconfigurable logic in 3D die stacks for test," in *Proc. IEEE 25th North Atlantic Test Workshop (NATW)*, May 2016, pp. 46–52. 59

[59] J. H. Lau and T. G. Yue, "Thermal management of 3D IC integration with TSV (through silicon via)," in *Proc. 59th Electronic Components and Technology Conference*, May 2009, pp. 635–640. 59

[60] A. Chandra and R. Kapur, "Bounded adjacent fill for low capture power scan testing," in *Proc. 26th IEEE VLSI Test Symposium (VTS)*, April 2008, pp. 131–138. 62

[61] Xilinx, "UG 474: 7 series FPGAs configurable logic block user guide," https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf, accessed: 2019-06-17. 67

[62] M. Loeblein, S. H. Tsang, Y. Han, X. Zhang, and E. H. T. Teo, "Heat dissipation enhancement of 2.5D package with 3D graphene and 3D boron nitride networks as thermal interface material (TIM)," in *Proc. 2016 IEEE 66th Electronic Components and Technology Conference (ECTC)*, May 2016, pp. 707–713. 72

[63] S. Mitra and K. S. Kim, "X-compact: an efficient response compaction technique for test cost reduction," in *Proc. International Test Conference (ITC)*, Oct 2002, pp. 311–320. 80

[64] D. K. Pradhan, S. K. Gupta, and M. G. Karpovsky, "Aliasing probability for multiple input signature analyzer," *IEEE Transactions on Computers*, vol. 39, no. 4, pp. 586–591, April 1990. 80

[65] W. H. Debany, M. J. Gorniak, D. E. Daskiewich, A. R. Macera, K. A. Kwiat, and H. B. Dussault, "Empirical bounds on fault coverage loss due to LFSR aliasing," in *Proc. IEEE VLSI Test Symposium*, April 1992, pp. 143–148. 80

[66] P. Alfke, "Efficient shift registers, LFSR counters, and long pseudorandom sequence generators," Xilinx, Tech. Rep., July 1996. [Online]. Available: https://www.xilinx.com/support/documentation/application_notes/xapp052.pdf/ 81

[67] J. Saxena, K. Butler, J. Gatt, R. Raghuraman, S. Kumar, S. Basu, D. Campbell, and J. Berech, "Scan-based transition fault testing - implementation and low cost test challenges," in *Proceedings. International Test Conference*, 2002, pp. 1120–1129.

[68] R. Chaware, K. Nagarajan, and S. Ramalingam, "Assembly and reliability challenges in 3D integration of 28nm FPGA die on a large high density 65nm passive interposer," in *Proc. IEEE 62nd Electronic Components and Technology Conference*, May 2012, pp. 279–283.

[69] M. Agrawal and K. Chakrabarty, "Test-cost optimization and test-flow selection for 3d-stacked ics," in *Proc. IEEE 31st VLSI Test Symposium (VTS)*, April 2013, pp. 1–6.

[70] S. Manoj P. D., J. Lin, S. Zhu, Y. Yin, X. Liu, X. Huang, C. Song, W. Zhang, M. Yan, Z. Yu, and H. Yu, "A scalable network-on-chip microprocessor with 2.5d integrated memory and accelerator," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 6, pp. 1432–1443, June 2017.

[71] E. Sperling, "Is the 2.5D supply chain ready?" *Semiconductor Engineering*, 2019, https://semiengineering.com/is-the-stacked-die-supply-chain-ready/ Accessed: 2019-11-26.

[72] A. Crouch, *Scan Architectures and Techniques*. Prentice Hall, 1999.

[73] P. Girard, L. Guiller, C. Landrault, S. Pravossoudovitch, and H. J. Wunderlich, "A modified clock scheme for a low power bist test pattern generator," in *Proceedings 19th IEEE VLSI Test Symposium. VTS 2001*, April 2001, pp. 306–311.

[74] S. Wu, L. Wang, L. Yu, H. Furukawa, X. Wen, W. Jone, N. A. Touba, F. Zhao, J. Liu, H. Chao, F. Li, and Z. Jiang, "Logic bist architecture using staggered launch-on-shift for testing designs containing asynchronous clock domains," in *2010 IEEE 25th International Symposium on Defect and Fault Tolerance in VLSI Systems*, Oct 2010, pp. 358–366.

[75] S. Gupta, "Improving System-On-Chip Test Networks For: Bandwidth, Security, and Power," PhD Thesis, Southern Methodist University, Dallas, TX, Spring 5-19-2018.

[76] S. Gupta, B. Bhaskaran, S. Sarangi, A. Abdollahian, and J. Dworak, "A Novel Graph Coloring Based Solution for Low-Power Scan Shift," in *2019 IEEE 37th VLSI Test Symposium (VTS)*, Apr. 2019, pp. 1–6.