Automation of Design For Test Flow-Focus on Faster Execution Cycle

Major Project Report - 3EC2401

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology
in
Electronics & Communication Engineering

(Communication Engineering)

By

Harsh K. Pandya

17MECC06

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2019

Automation of Design For Test Flow-Focus on Faster Execution Cycle

Major Project Report - 3EC2401

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology
in
Electronics & Communication Engineering

(Communication Engineering)

By

Harsh K. Pandya

17MECC06

Under the guidance of

External Project Guide:

Mr. Aditya Joshi,

SoC Design Engineer,

DFT Team-CRCG,

Intel Tech. India Pvt. Ltd., Bengaluru

Internal Project Guide:

Dr. Y. N. Trivedi,

Professor in EC Engineering,

Institute of Technology,

Nirma University, Ahmedabad

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2019

Declaration

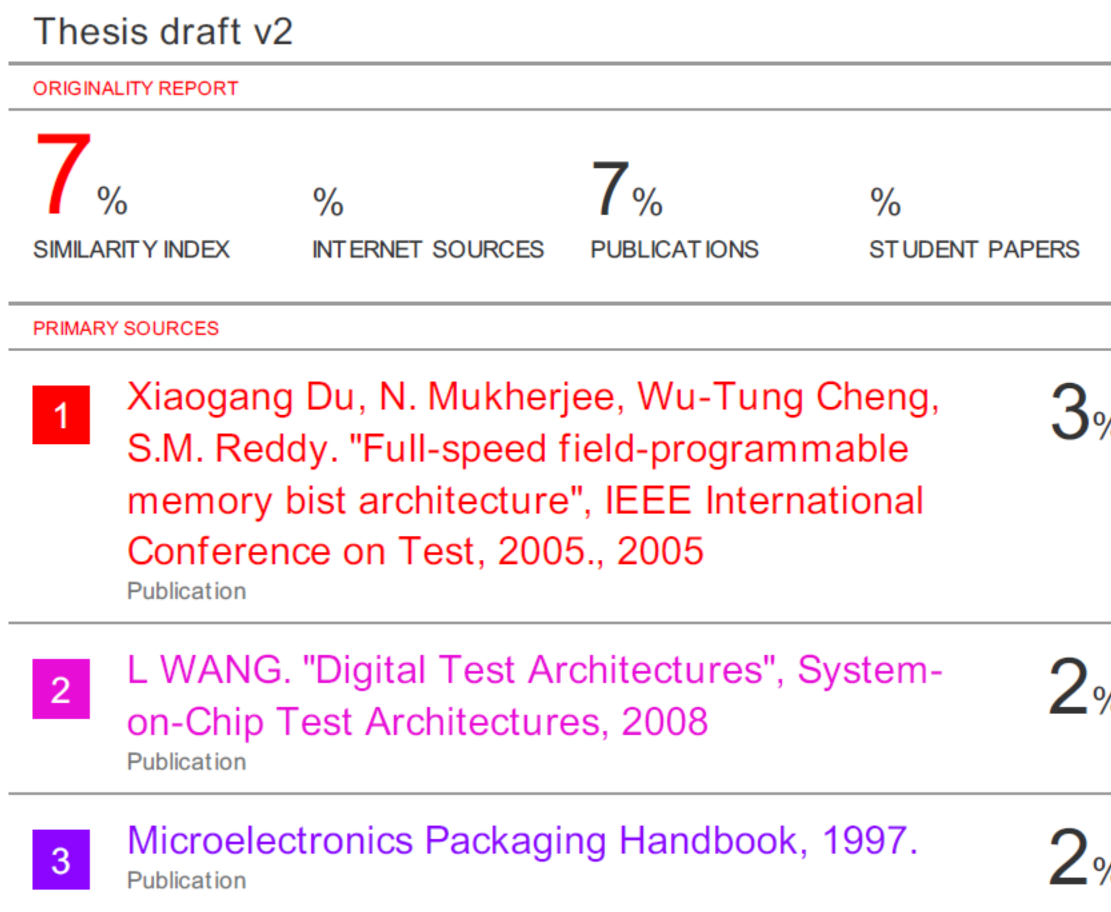
This is to certify that

1. The thesis comprises my original work towards the degree of Master of Technology in Communication Engineering at Nirma University and has not been submitted elsewhere for a degree.
2. Due acknowledgment has been made in the text to all other material used.
3. Complete document with its contents and its results has been verified and reviewed from the Intel.

- Harsh Pandya

17MECC06

Plagiarism Check


Thesis draft v2

ORIGINALITY REPORT

7%	%	7%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Xiaogang Du, N. Mukherjee, Wu-Tung Cheng, S.M. Reddy. "Full-speed field-programmable memory bist architecture", IEEE International Conference on Test, 2005., 2005 Publication	3%
2	L WANG. "Digital Test Architectures", System-on-Chip Test Architectures, 2008 Publication	2%
3	Microelectronics Packaging Handbook, 1997. Publication	2%

Figure 1: Plagiarism Checked at Turnitin.com

Dr. Y. N. Trivedi	Harsh K. Pandya
Internal Project Guide	17MECC06
Sign:	Sign:

Disclaimer

"The content of this thesis does not represent the technology, opinions, beliefs, or positions of Intel Technology India Pvt.Ltd., its employees, vendors, customers, or associates".

Certificate

This is to certify that the report entitled " **Automation of Design For Test Flow-Focus on Faster Execution Cycle** " submitted by **Harsh K. Pandya (Roll No: 17MECC06)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Communication Engineering of Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this training, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Dr. Y. N. Trivedi
Internal Guide,
Professor in EC Engineering,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Y. N. Trivedi
Program coordinator,
Professor in EC Engineering,
Institute of Technology,
Nirma University, Ahmedabad.

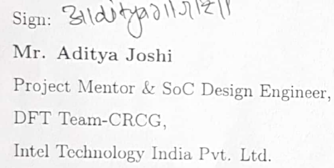
Dr. D. K. Kothari
Professor and Head,
EC Engineering Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Alka Mahajan
Director,
Institute of Technology,
Nirma University, Ahmedabad.

vii

Certificate

This is to certify that the report entitled " **Automation of Design For Test Flow-Focus on Faster Execution Cycle** " submitted by **Harsh K. Pandya (Roll No: 17MECC06)**, towards the submission of the Project for requirements for the award of degree of Master of Technology in Communication Engineering of Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Sign: 

Mr. Aditya Joshi

Project Mentor & SoC Design Engineer,

DFT Team-CRCG,

Intel Technology India Pvt. Ltd.

Date: 10/5/2019

Place: Bengaluru

Figure 2: Company Certificate

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Dr. Yogesh Trivedi**, Professor, Electronics and Communication Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance and continual encouragement throughout this work. The appreciation and continual support he has imparted has been a great motivation to me in reaching a higher goal. His guidance has triggered and nourished the intellectual maturity that I will benefit from, for a long time to come.

I sincerely thank **Dr. D. K. Kothari**, Hon'ble Head of Electronics and Communication Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his kind support and providing basic infrastructure and healthy research environment.

I heartily express my sincere gratitude towards **Mr. Aditya Joshi** (SoC Design Engineer, DFT Team, Intel) and **Mr. Hemang Mistry** (Reporting Manager, DFT Team, Intel) for supporting me with all the resources and trusting me for this endeavor. I am also thankful to **Mr. Ritesh Rampal** and **Mr. Madhusudan G.S.** for their constant support during various assignments that had been given to me.

Last but not the least, I would thank my wife and my parents for their unconditional love and support during the project work.

Harsh K. Pandya
17MECC06

Abstract

This thesis focuses on the Design For Testability (DFT) part of the Pre-Silicon designing stage of an RF chip for client platforms. DFT is the process of inserting some standard test modules for checking an RF chip's testability. The process of creating such modules involves designing of various modules like Scan, BIST etc. The thesis further goes into the process of generating memory views, creation of memory-wrappers using various third-party tools being performed on the Linux environment. The process of DFT includes numerous steps like Architecture design, RTL coding according to the given requirements, and Verification followed by Synthesis, Gate-level simulation and DFT logic insertion. It also includes integrating various Intellectual Properties (IP) from different Business Units (BU) or workgroups to be put onto the System on Chip (SOC), into which embedded memories are a part of them. The thesis explains the scope of embedded memories, memory wrappers and their generation process using a shell script operated Library generator and parallel flow commands respectively. It also gives a brief knowledge on introducing some of the automation into the pflow steps, which requires the least possible user input and can reduce the number of steps for the process.

Keywords: Automation, Embedded Memories, MBIST, Wrapper

Abbreviations

ASIC Application Specific Integrated Circuit

ATE Automatic Test Equipment

ATG Automatic Test Generation

ATPG Automatic Test Pattern Generation

ADFT Automatic Design for Testability

BISR Built-in Self Repair

BIRA Built-in Repair Analysis

BIST Built-in Self Test

BU Business Units

CASP Concurrent Autonomous Self-Test using Stored Patterns

CUT Circuit Under Test

DC Design Compiler

DFT Design For Testability

DUT Device Under Test

DVE Discovery Visual Environment

DRAM Dynamic Random Access Memory

ECC Error Correcting Code Memory

EDT Embedded Deterministic Test

EEPROM Electrically Erasable Programmable Read-only Memory

ETM Element-Test and Maintenance

FP-BIST Field Programmable BIST

FSM Finite State Machine

GTE General Telephone & Electronics Corporation

IP Intellectual Properties

JTAG Joint Test Action Group

MBIST Memory Built-In Self Test

MISR Multiple-Input Signature Register

MOS Metal Oxide Semiconductor

OCC On-Chip Clock Control

RTL Register Transfer Layer

SDC Synopsys Design Constraints

SOC System on Chip

SPRAM Single Port RAM

SRAM Static Random Access Memory

TAP Test Access Port

TCK Test Clock

TDI Test-Data In

TDO Test-Data Out

TM Test and Maintenance

TMS Test Mode Signal

TPRAM Two Port RAM

TSDB Tessent Shell Data Base

VAST Virtualization-Assisted Concurrent Autonomous Self Test

VCS Version Control System

VHSIC Very High Speed Integrated Circuits

Contents

Declaration **iii**

Plagiarism Check **iv**

Disclaimer **v**

Certificate **vi**

Company Certificate **vi**

Acknowledgements **viii**

Abstract **ix**

Abbreviations **x**

List of Figures **xiv**

1 About the Company **1**
1.1 Intel Ethics and Work Culture 1

2 Intel Mobile Modems **2**
2.1 Intel Mobile Modems and Modules Overview 2
2.2 Intel Eyeing 5G and More 2
2.3 An Overview on 5G Modem 2
2.3.1 Key Features of 5G Modem 3

3 Introduction to Domain **4**
3.1 Design for Testability 4
3.1.1 Basic Design of System-on-Chip 4
3.1.2 Basic ASIC Design Flow 5
3.2 Impact of Semiconductor Technology on Testability 6
3.3 Evolution of DFT 6
3.3.1 Standards for DFT 7
3.3.2 Automation in DFT 8

4 Practical Exposure **10**
4.1 Trainings Undergone 10

5 Memory Generation and MBIST **12**
5.1 Embedded Memories Evolution 12
5.1.1 Memory Generation 14
5.2 MBIST 15
5.3 Memory Wrappers 15
5.3.1 Parallel Flow 16
5.3.2 Scan-Insertion 17
5.4 Memory Repair 19
5.4.1 Advancements in Memory Repairs 22
5.5 Automation Introduced 25
5.6 Conclusion 26
5.7 Possible recommendations 26

6 Appendix **27**

Appendix **27**

Bibliography **29**

List of Figures

1 Plagiarism Checked at Turnitin.com iv
2 Company Certificate vii

3.1 Basic SoC model 4
3.2 Basic ASIC Design Flow 5
3.3 The outcome of scale introduced on Gate and Wire Delay 6
3.4 Evolution of DFT realized for testing 7
3.5 Chip Architecture supporting IEEE 1149.1 JTAG standard 8
3.6 Silc design cycle with ADFT components 9

5.1 First 20 years of stand-alone MOS memory characteristics 12
5.2 Embedded MOS memory characteristics 13
5.3 MBIST Wrapper 16
5.4 MBIST Shell 17
5.5 Case: Optimal Spare Allocation 20
5.6 BISR Architecture 22
5.7 FP-BIST Top-level Architecture 23
5.8 Automation Script Output 25

6.1 Schematic for Verilog example 27
6.2 Chip realization for Verilog example 27
6.3 Simulation by DVE Tool 28

Chapter 1

About the Company

Intel is world's second largest valued semiconductor chip markers headquartered in Santa Clara, California and is founder of x86 series of microprocessors founded in most of personal computers. Along with processors, Intel also manufactures mother board chipsets, IC's, Graphics' chips, Network interface chips etc. Intel has constantly changed its role as the generation changed, from memory manufacturing, to processor manufactures to now become a data-centric company. Intel has also got its capabilities in the field of Machine / Deep Learning, Programmable solutions, Memories, Communication chips' design,FPGA and IOT domains. Intel is on a paradigm-shift towards enabling devices to Process the data,Transport it and get Delivered to the end user. Intel workforce consists of a diversified portfolio. It had 8.5% increment in terms of Women representation. 31.4% increase in African American representation etc. to name a few in terms of diversity.

1.1 Intel Ethics and Work Culture

Intel's mission to utilize Moore's Law is to bring smart, connected devices to every person on Earth. Being a part of a bold and curious team of inventors and problem-solvers who continue to create some of the most astounding technology advancements and experiences in the world. Guidance and Direction comes through innumerably layers of management/leadership, to employees. All are guided by 6 core values namely-

- Customer orientation
- Results training
- Risk taking
- Great place to work
- Quality
- Discipline [1]

Chapter 2

Intel Mobile Modems

2.1 Intel Mobile Modems and Modules Overview

Intel has been the biggest player of the microprocessor technology and is a leading innovator in CISC architecture based processors. Almost 80% of the laptop and desktop computers use Intel's microprocessors and thus is the market leader in microprocessors. Intel has adopted Moores law applied to silicon technology which states that the number of transistors that can be placed on a chip doubles every eighteen months. Designed for smartphones, tablets, PCs, and Internet of Things (IoT) devices; Intels low-power modems and SOC are among the industrys most compact and cost-effective mobile solutions.

2.2 Intel Eyeing 5G and More

Based on the millimeter waves' unexplored spectrum, Intel is trying to create a holistic ecosystem which connects all the devices powered by Intel with a centralized and secured cloud. Intel's CRCG (Convergence, RF and Connectivity Group) is a part of ICDG (Intel Communications and Devices Group) which is responsible for 5G technology advanccement.[2]. Intel Mobile Communications provides cost-efficient 2G and 3G single-chip platforms for ultra-low-cost mobile phones and entry-level smartphones, to leading-edge 3G and 4G slim modem and radio frequency (RF) solutions for smartphones and tablets.[3]

2.3 An Overview on 5G Modem

As wireless spectrums are on the verge of exhaustion for 4G technology, it's essential that there should be a technology which not only finds a new spectrum; but also it should be focused on the higher amount of data rate as the demand for faster communication

is risen. It's estimated that by 2035, the 5G industry value chain will be worth $ 2.5 trillion. To deploy 5G, it's important to deploy the cell sites and enhance the infrastructure which requires local authority's role too. Intel has run trials on early 5G networks with the collaboration with partners for the Winter Olympics held in Korea. Intel has developed a 5G modem which supports various high speed data activities. Also, Intel has an exclusive platform to test and develop 5G related development and support. [4]

2.3.1 Key Features of 5G Modem

- Worlds first single chip to support 5G operation in both sub-6 GHz and mmWave bands.
- Expected speeds exceeding 5 Gbps, hundreds of MHz of aggregated bandwidth and ultra-low latency.
- Pairs with the worlds first 5G sub-6 GHz RFIC and the mature 28 GHz 5G mmWave RFIC.
- Supports key 5G NR technology features, including low latency frame structure, advanced channel coding, massive MIMO and beamforming.
- Pairs with LTE modems such as Intels XMM 7360 LTE modem to provide 4G fallback, and 4G/5G interworking. [5]

Update: On 16th April 2019, Intel announced exit from 5G smartphone modem business. However, 5G will be continued to be the strategic priority as the management emphasized. [6]

Chapter 3

Introduction to Domain

3.1 Design for Testability

The DFT domain broadly focuses on four parameters: Power, Performance, Area and Schedule. Design for Testability includes Controllability, Observability, and Predictability. Most of the DFT techniques deal with enhancing the said parameters. Normally, in most cases, the amount of a circuit's controllability and observability is measured in terms of whether the tests are generated randomly; or generated by using any Automatic Test Generation (ATG) algorithms. Most DFT techniques deal with the re-synthesis of the current design, or an inclusion of the test-hardware in the same. So the said approach requires changing the factors like area, I/O pins, and circuit delay. [7]

3.1.1 Basic Design of System-on-Chip

The basic design for SOC model [8] is shown below -

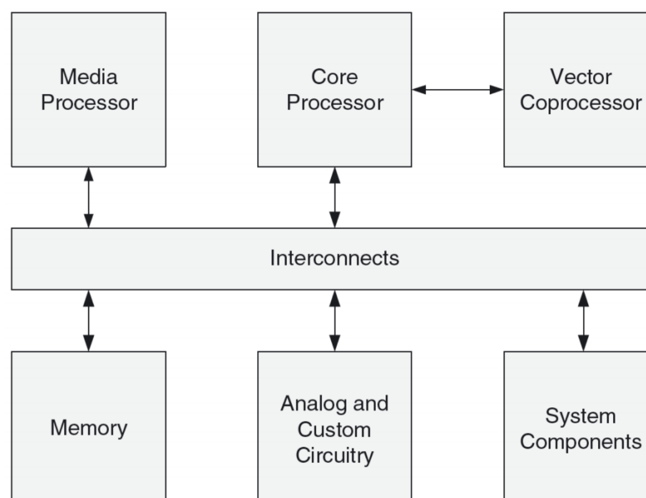


Figure 3.1: Basic SoC model

Generally, an SoC must have at least one processor core. But nowadays, mobile computing SoCs consist of various blocks like Core processor, various high density single and dual-port memories, on-chip caches, RF capabilities and often digital camera hardware as well as firmware.

3.1.2 Basic ASIC Design Flow

The basic Application Specific Integrated Circuit (ASIC) design flow provides various viability and optimization problems that require efficient and efficacious algorithms to be designed taking into account the increasing complexity being introduced in VLSI designs while there is an era of ultra-deep submicron technologies.[9] The basic ASIC design flow for SoC architectures is shown as follows:[10]

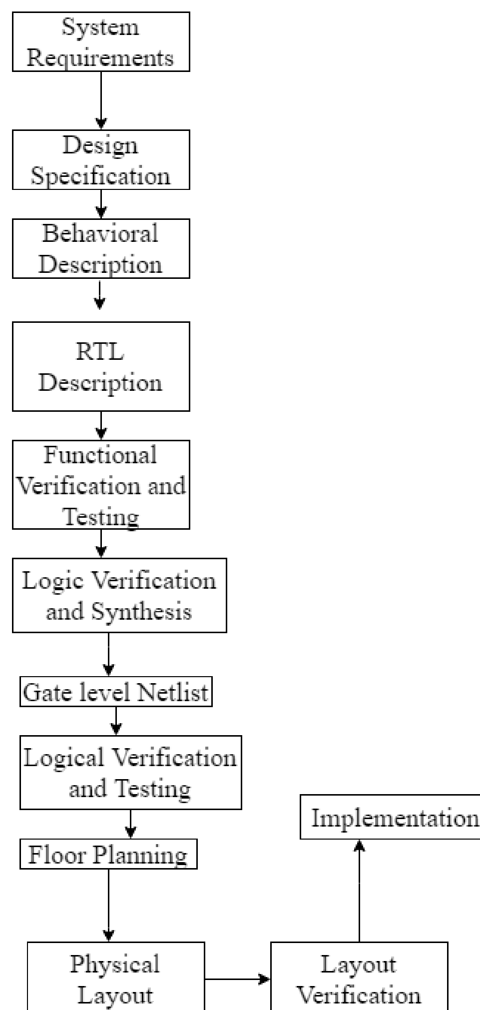


Figure 3.2: Basic ASIC Design Flow

3.2 Impact of Semiconductor Technology on Testability

With 250-180 nanometer technology, the focus for DFT changes rapidly. The area estimation due to the addition of test logic does not affect the chip area. This is because the technology is shifted to the deep sub-micron level. Also, the timing of the design is dominated by interconnect delays. The delays are optimized while using the automation in the test technology. As the scale of the technology increases, electromagnetic coupling also got incremented. Test technology needs to take into account this fact for maintaining the quality of testing for delay sensitive defects and their connected tests. The amount of delay is shown in the figure shown.

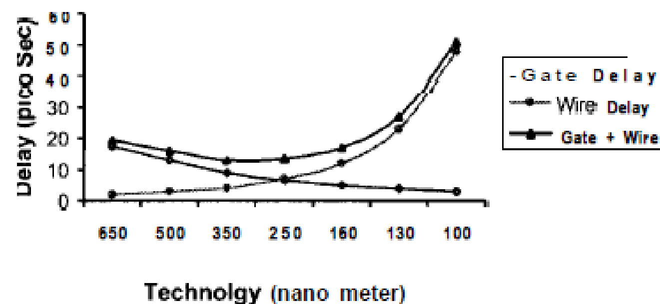


Figure 3.3: The outcome of scale introduced on Gate and Wire Delay

There are two types of delay defects being encountered while chips are being tested:
1. Path Delay testing, and
2. Transition fault testing.

Path delay testing aims to the defects that are distributed across the chip in the manufacturing process. Transition fault testing is used for spotting the defects that instigate a delay at the location where the defect occurs.

Test structures can be used to create foreseeable flows for successful testing of the designs and they ought to be used to reduce the test-cost. Built-in Self Test (BIST) structure is one of the example of such a DFT technique. [11]

3.3 Evolution of DFT

The semiconductor industry heavily relies on two techniques for testing digital circuits: Scan and Logic BIST. Scan is implemented by first replacing all selected storage elements

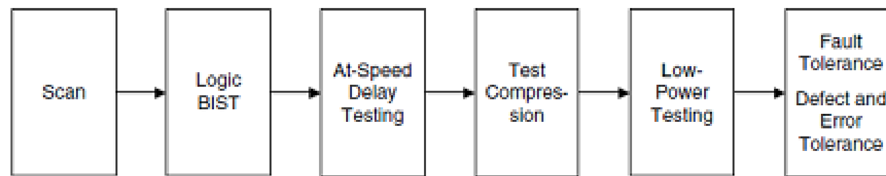


Figure 3.4: Evolution of DFT realized for testing

of the digital circuit with scan cells and then joining them into one or more shift registers, called as 'scan chains', to provide them with external access. Some of the drawbacks with the scan approach are:

1. Traditional test schemes using Automatic Test Pattern Generation (ATPG) software to check single faults have become a bit expensive in terms of cost, and

2. Sufficiently high fault reportage for these deep submicron/nanometer VLSI designs is hard to maintain from the chip level to the board and to system levels. To overcome the said problems, scan approach is combined with the Logic BIST approach. Generally, pseudo-random patterns are applied to the Circuit Under Test (CUT) while their test responses are condensed in a Multiple-Input Signature Register (MISR). [12]

3.3.1 Standards for DFT

To consider broad-level testing, several standards have been formulated. The prime objective to develop these standards was to decrease the complexity and cost attached with the testing. Some of these initiatives are known as the Joint Test Action Group (JTAG) Boundary-Scan standard, the VHSIC Element-Test and Maintenance (ETM) - Bus standard, the VHSIC Test and Maintenance (TM) - Bus standard, and the IEEE 1149.1 Testability Bus Standard. The primary reasons for using boundary scan are to allow for proficient testing of board interconnect, and to smooth isolation and testing of chips either via the test bus or by built-in self-test hardware.

With boundary scan, chip-level tests can be reused at the board level. There are several major components associated with IEEE 1149.1 viz.

(1) the materialized structure of the test-bus and how it can be interconnected to chips,

(2) the protocol associated with the bus, and

(3) the on-chip test bus circuitry connected with a chip. The latter includes the boundary-scan registers and the Test Access Port (TAP) controller which is a Finite State Machine

(FSM) that decodes the situation of the bus.

The general architecture for the IEEE 1149.1 is shown as follows.

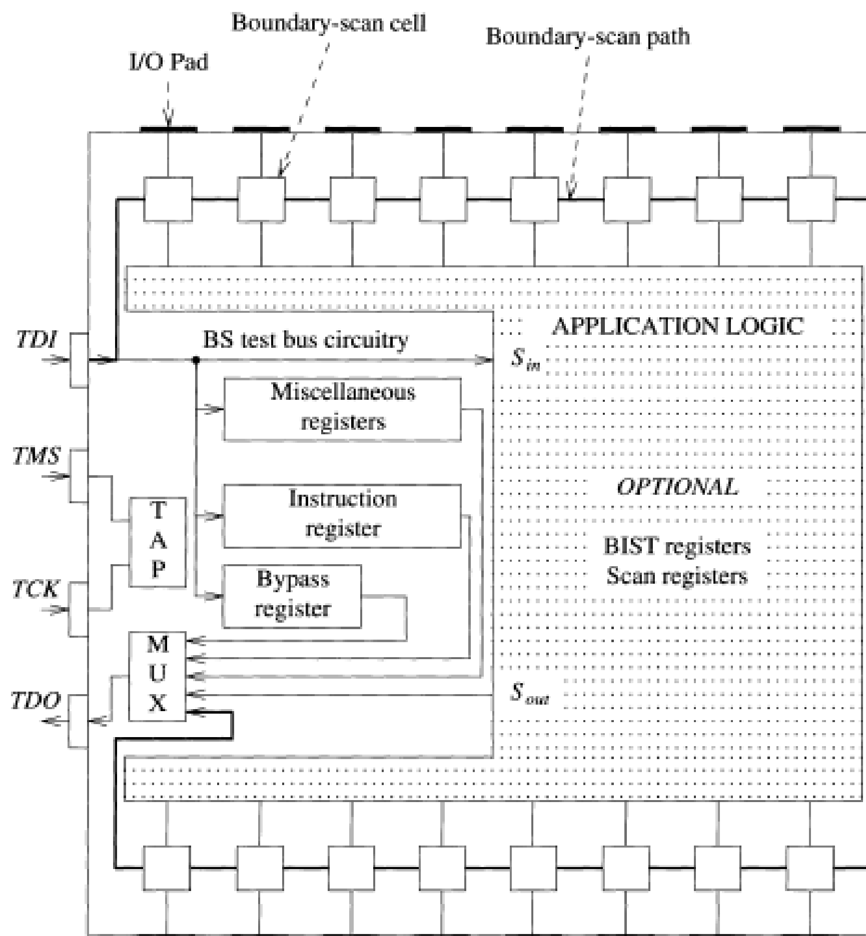


Figure 3.5: Chip Architecture supporting IEEE 1149.1 JTAG standard

As shown in the figure, this circuitry may include DFT or BIST hardware. The test-bus circuitry is also named as the bus-slave which consists of boundary-scan registers, a 1-bit bypass register, an instruction register, other registers and the TAP. The boundary-scan bus consists of four lines, namely Test Clock (TCK), Test Mode Signal (TMS), Test-Data In (TDI) line, and Test-Data Out (TDO) line. [13]

3.3.2 Automation in DFT

Silicon compilation is being used as an alternative for the traditional IC design process although not considering testability issue. So, automatic DFT software is needed for the same.[14] Therefore, researches for testability have found two applications. The first among both is to help the designer in modifying a circuit to make it more testable.

Testability usually involves measures like reducing test generation costs, increasing fault coverage, making the circuit more initializable, or removing redundancy in this scenario. The later one suggests to use these measures within a test generation program for reducing the CPU time.[15]

A behavioral silicon compiler, named as Silc, developed at General Telephone & Electronics Corporation (GTE) Laboratories (Acquired by Bell Atlantic in 2000, later combined company was named as Verizon) was able to create fabrication masks for full-custom VLSI chips which were being used in telecommunications. User input in the Silc is the specifications of a chip's behavior which is coded in the language like Lisp while considering the FSM that can function in parallel and can communicate either synchronously or asynchronously.

Silc's circuit design has many inferences for Automatic Design for Testability (ADFT) system. The drawback with the earlier ADFT approaches was their application of a single DFT technique to the entire circuit. However, a modular DFT approach was proposed keeping in mind the Silc design cycle along with ADFT components as shown below:[16]

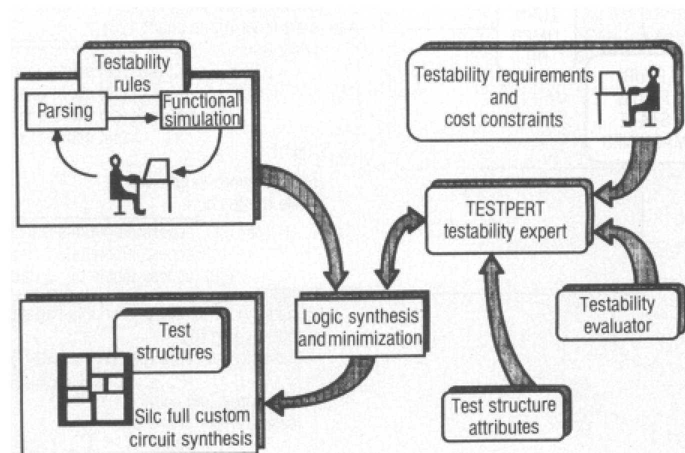


Figure 3.6: Silc design cycle with ADFT components

Chapter 4

Practical Exposure

4.1 Trainings Undergone

- The basic understanding on Verilog, Perl, VCS and DVE tools has been acquired.
- Practical exposure included study of Memory wrappers' data-sheet prepared for the given specifications.
- Also, memory library files were used to generate the definition-file for the required tasks using given set of parameters.
- Generation of testbenches and their simulations were carried out. Perl was used to automate the processing of Verilog coded file. (Status: In progress)
- Perl code for reading contents from a sample file:

```
#!/usr/bin/perl
my filename = "sample1.txt";
open(FH, " < ", $filename)ordie$!;
while(< FH >)
{
print$_;
}"
close(FH);
```

- Verilog Sample Example:

```
module example2 (A,B,C,D,E,F,Y);
input A,B,C,D,E,F;
output Y;
wire t1,t2,t3;
```

```
nand #1 g1 (t1,A,B);
and #1 g2 (t2,C,D);
nor #1 g3 (t3,E,F);
nand #2 g4 (Y,t1,t2,t3);
endmodule
```

Testbench:

```
module testbench2;
reg A,B,C,D,E,F;
wire Y;
example2 DUT(A,B,C,D,E,F,Y);
initial
begin
$monitor ($time, "A=%b, B=%b, C=%b,D=%b,E=%b,F=%b,Y=%b",A,B,C,D,E,F,Y);
#5 A=1;B=0;C=0;D=1;E=0;F=0;
#5 A=0;B=0;C=1;D=1;E=0;F=0;
#5 A=1;C=0;
#5 F=1;
#5 finish;
end
endmodule
```

The simulated results are shown in the appendix.

Chapter 5

Memory Generation and MBIST

5.1 Embedded Memories Evolution

As the scope of the work assigned was focused on the embedded memory generation, a brief evolution will be helpful to understand the memory repair strategies, too. The earlier version of memories, from 1980s to 1990, the ideal Metal Oxide Semiconductor (MOS) memories had small cell size, better array efficiency, sufficient performance, noise and soft error resistance, and reached an external I/O standard. It consisted of three memory types viz. Static Random Access Memory (SRAM)s, Dynamic Random Access Memory (DRAM)s, and Flash Electrically Erasable Programmable Read-only Memory (EEPROM)s, which were used in different applications as mentioned below.

	SRAM		DRAM		Flash/EEPROM	
	1980-1990	1990-2000	1980-1990	1990-2000	1980-1990	1990-2000
Read speed	fast (ns)	faster (ns)	moderate (ns)	fast (ns)	moderate (ns)	moderate (ns)
Write speed	fast (ns)	faster (ns)	moderate (ns)	fast (ns)	very slow (ms,s)	very slow (ms,s)
Non-volatile	no	no	no	no	yes	yes
Cell size	4	6	1.5	1.5	1	1
Cell type	NMOS	CMOS	planar	vertical	NOR	NOR & NAND
Density	low	low	high	high	high	high
Supply voltage	5 V	3.3/2.5 V	5 V	3.3/2.5 V	5 V	3.3/2.5 V
Write voltage	5 V	3.3/2.5 V	5 V	3.3/2.5 V	18 V	12 V
Mask adders	none	none	n.a.	8–11	n.a.	8–11
Standardization	I/O	spec.	I/O	spec.	I/O	spec.
Application	cache	PDA	server	PC	BIOS	cell phone

Figure 5.1: First 20 years of stand-alone MOS memory characteristics

By 1990 to 2000, memories started to have significant amounts of logic integrated onto the chip. Some embedded DRAM and Flash appeared but were obstructed by the historical dissimilarity of the memory and logic technologies. This move to adding logic on the memory chip was driven by several factors. Sub-micron geometries have both increased logic speed requiring faster memories, and minimized cell size providing space

for on-chip logic. From 2000 to 2005, Embedded memories were being started to be used because of the potential of integrating large subsystem sections on the chip; thus becoming a large part of the chip. Characteristics of embedded memory are different from a stand-alone memory. Boundary scan (JTAG), BIST and Built-in Self Repair (BISR) bring test on chip and Error Correcting Code Memory (ECC) on chip decreases soft error problems. There is an important measure nowadays for the embedded memories, which is their compatibility with the CMOS logic process. On one hand, specialized memory processes increase the cost of the logic chip; and on the other hand, Planar DRAM cells and single polysilicon flash memory cells result in comparatively sizeable cell size. A current scenario for the given embedded memories is given below:

	SRAM	**DRAM**	**Flash/EEPROM**
Random read speed	very fast	fast	moderate
Soft error	ECC	ECC	n.a.
Interface	custom	custom	custom
Mask adders to CMOS	none	0–8	0–11
Supply/write voltage	1.8/1.1 V	1.8/1.1 V	1.8/1.1 V
Verified macro IP	yes	yes	yes
Test	JTAG, BIST	BIST, BISR	external

Figure 5.2: Embedded MOS memory characteristics

These memories are made up from either a large number of 'hand-optimized' blocks or smaller memories generated by compilers. They take a large area in the SOC. There are some benefits as well as drawbacks for the embedded memories which are mentioned below. [17]

<u>Benefits</u>:	<u>Drawbacks</u>:
1. Improved performance	1. Increment in development complexity
2. Lower power consumption	2. They need to be area and yield optimized
3. Package cost reduction	3. Higher mask cost
4. Exact granularity and organization	4. Decreased flexibility and extendibility
5. Reduction in overall cost	5. Overall test cost and tester requirements needs to be taken into account.

As density of these embedded memories increases, manufacturing yield also increases up to a level where it can't be accepted. So, redundancy is added by designers in the circuits.

Yield prediction is very vital because too much redundancy wastes the silicon area,and too little redundancy leads to poor yield.[18] Although the memories have certain advantages, there are faults encountered too while testing the chip. So, memory repair task is also incorporated along with the BIST. According to [19], there are following repair strategies:

- Hard: Requires permanent storage for storing repair information even after power is turned off.
- Soft: Repair signature generated at every power-up, no requirement for storing the repair information.
- Combination: This method provides good elements from soft and hard repair.
- Cumulative: In this, a memory stores the repair information in a non-volatile fuse box which uses the prior saved data at the starting of next test and repair cycle.

5.1.1 Memory Generation

Multi-port RAMs are important for high-performance parallel computation systems.There can be a requirement of Single Port RAM (SPRAM) and Two Port RAM (TPRAM)s for the given specification. As per the spec, memory models are generated which include Virage models consisting various Mux blocks. These views are Verilog files which then be integrated with the main project configuration.The steps for generating the memory from particular specifications is as follows:
The library generation refers to the process of creating a list of customized memory views in the Synopsys-Tessent tool.

- The views represent the memories defined in the .csv file.
- The .csv file consists of a list of memories required.
- The process follows with changing a definition (.def) file according to the .csv file's respective parameters.This .csv is realized from the Register Transfer Layer (RTL) design.
- A libgen script is run which takes the .def file as an input and creates memory views according to it.

- The memory views are represented as a Verilog (.v) file having all the specifications for the Memory Built-In Self Test (MBIST) block.

5.2 MBIST

Embedded memories are very vital part of any sub-micron SoC. As the amount of memories are being increased day-by-day on the SOC, it's important to test memories beforehand the manufacturing of the silicon. MBIST is one of the most widely used techniques that tests memories and faults if any. MBIST also plays an important role for the testing of memories post-manufacturing. The memory BIST consists of a standard memory BIST controller and the redundancy logic interface, which provides the following signals: [20]

- Expected data which is used to compare the test results from the memory.
- Fail address which stores address of faulty data.
- Fail signal which is used as a write enable for the redundancy logic.

MBIST block has some advantages so far as testing is concerned:[21]

- Reduction in Design complexity as there is no need for managing direct access of memories from top level.
- Test costs get reduced due to decremented test time and test resources.
- It gives a possibility to run different algorithms on memories.
- It can also be used for burn-in testing of memories.
- Repair data calculation is automatically performed by BIST.

5.3 Memory Wrappers

MBIST can be functionally implemented as a wrapper over the memories to ensure efficient functional testing.The latest SoCs contain hundreds of memories. Testing all the memories in these SoCs sequentially would take a long time. If memory BIST logics were individually added to these various memories, the area overhead would be very high. In order to avoid that, memory level wrapper is applied to reduce the test-time as well

as complexity as mentioned in the MBIST section. The time period to test memories depends upon:

- Specific algorithm needed to be run for respective memory so as to decide number of cycles for BIST.
- Size of the largest memory among all of the memories which needs BIST to be run.[21]

MBIST as a wrapper for memory cells is shown below:

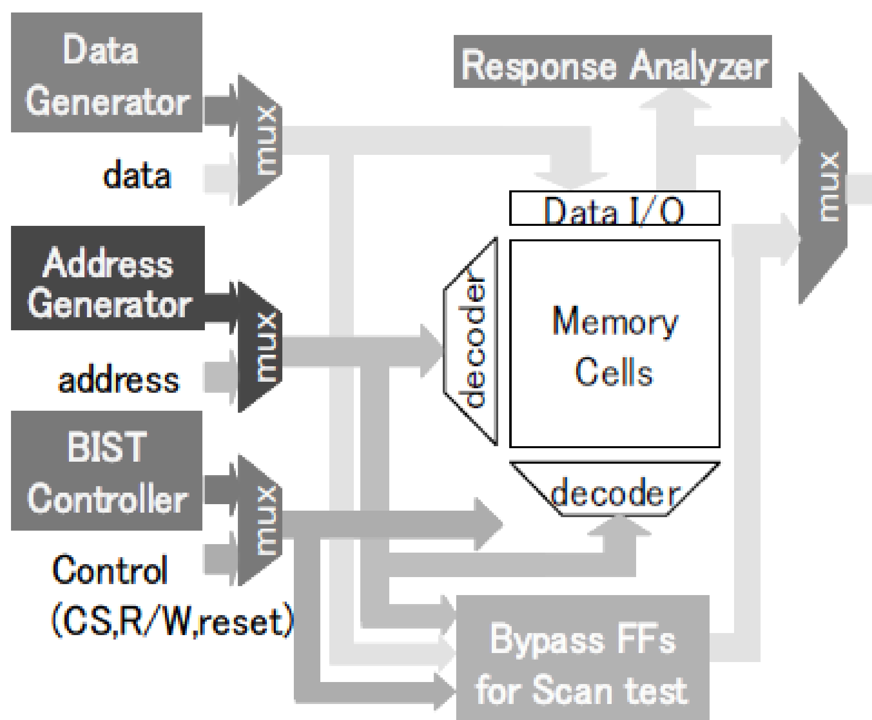


Figure 5.3: MBIST Wrapper
[22]

5.3.1 Parallel Flow

Parallel flow (often called as Pflow) is an MBIST implementation concept, in which MBIST RTL is delivered as an IP to the project specification. General implementation steps for Pflow are as follows:

- Making design MBIST ready
- Generation of mbistshell and its delivery: Here the input required is the memory list.

- Combining mbist and memory design: mbistshell is instantiated with the memory, till 2015, TDI and TDO were being wired manually by RTL designer.

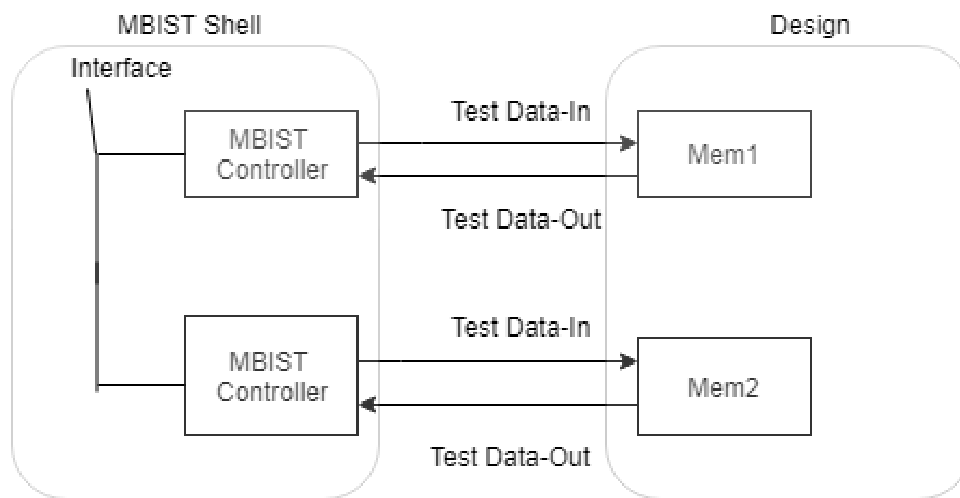


Figure 5.4: MBIST Shell

The mbistshell contains one dft bus-in and dft bus-out signals for every memory.[23] Steps ahead for mbistshell:

- The mbistshell and dft wrapper are connected via dft bus.
- Designer incorpotes the DFT Wrapper in the design.
- Bus connection either manually or using any automated scripts
- Verification of mbistshell and dft wrapper at the generation stage
- Same Test-benches can be used on post-integration stage also. There is a need to adapt the environment for inclusion of the Tessent test-benches if required.
- Synopsys Design Constraints (SDC) are generated at mbistshell level, and they can be re-targetted using Pflow utilities to a higher hierarchy where synthesis partition exists.

5.3.2 Scan-Insertion

The file containing the information about Tessent DFT logic is used with the given template to perform scan insertion. This is particularly useful when the user performs

Tessent DFT logic insertion (e.g. Tessent MemoryBIST) in RTL. However, the Scan-insertion is done via a third party tool i.e.Tessant Scan. If Tessent Scan is used, then the user need not use any other command as it's done by Tessent automation infrastructure. The Testcase for inserting a DFT logic has following steps:

- Inserting MBIST logic in RTL.
- Insert Tessent's TestKompress™ and Mentor Graphics' On-Chip Clock Control (OCC)® using DFTSpec. in RTL.
- Synthesize the RTL design and RTL DFT logic together into a netlist.
- Generate template, populate with tool commands to perform scan insertion.

In this testcase, the Tessent Shell Data Base (TSDB) is used to store the outputs of each DFT insertion step in the directories which are tagged with a unique design-id provided by the user. This design-id simplifies the forward feeding of data when incremental DFT logic is performed.

Explanation for each steps:

Step-1:

The MBIST logic is generated and inserted in the RTL design. The design-id used in this step is let's say rtl1.

Step-2:

In this step, the Embedded Deterministic Test (EDT) and OCC are generated and inserted using DFTSpec in the RTL design with MBIST from step 1. The design-id used in this step is let's say rtl2.

Step-3:

The DFT inserted design from step-2 is synthesized in this step. The file list to be synthesized is written in step-2, which is an input to the Design Compiler (DC). The thing to be noted here is, this step is performed outside the Tessent flow, so the user input is required in saving the netlist into the TSDB. The netlist is saved at the location provided by the user in DC synthesis script.

Step-4:

As stated earlier, if Tessent Scan is used for the scan-insertion, Tessent automation infrastructure allows the user to automatically read the information about previously inserted

Tessent DFT logic. This is done simply by reading the output netlist from step 3. The template generated to include Tessent Scan commands and executed in Tessent Shell as shown in the 'dofile'.

A tool like DC is unaware of the Tessent environment so it can't interpret the information related with the existing DFT logic from the files present in TSDB. The generated template helps the user in manually adding the commands to provide the tool with the information about previously inserted DFT logic. After adding the commands, the template can be executed in DC shell environment with other required configurations to perform scan insertion. [24]

5.4 Memory Repair

It's important to understand the memory repair process while using third party tools for its generation as well to generate wrappers. Preparing a repair methodology often requires combining IP from memory suppliers, automation from DFT providers, and data from respective foundries. The memory repair process has 03 basic components:

1. Test,
2. Repair Analysis, and
3. Repair Delivery.

A brief explanation is given below for each stage of the process.

1. **Test:**

   As discussed in Section 5.3, the MBIST consists of an on-chip engine, which is placed next to each embedded memory. It writes algorithmically generated test-patterns to the memory and then reads these patterns back to discover and possibly log any defects.

   The memory BIST engine is typically designed to generate patterns based on a predetermined memory test algorithm encoded in a finite state machine.

   As memory densities are growing rapidly, it's difficult to predict any defect mechanisms, which ultimately lead to a challenge for testing them. So, in order to get a solution for these issues, some MBIST solutions provide programmable BIST engines.These engines would incorporate the new as well as pre-determined defect mechanisms.

But, drawback here is these engines are larger than their conventional counterparts. So, it can be said that these new engines are to be used when there is a need.

2. **Repair Analysis:**

   Repair Analysis can be performed in two ways: On-Chip and Off-Chip. In the Off-chip approach, all memory failures are logged and this fail-data is post-processed offline. The major drawback here is it has a very large increment in test time.

   Due to this disadvantage, a large part of today's repair approaches uses on-chip repair capability. It's also known as Built-in Repair Analysis (BIRA). In BIRA, there is no need for logging the fail data as its engine/circuitry checks for the fail-data which is coming out from an associated BIST controller while testing is being run.

   By the end of the memory test, the BIRA engine has determined the spare element allocation required for repairing of the chip. To get BIRA succeed, there is an ability to find spare allocations. If spare rows or columns are to be found, then solution is easier as the defective blocks need be replaced.

   But, it becomes more complex when both spare rows and columns are available. A case explaining the said scenario is shown below where fig.(a) contains 2 spare rows and one spare column and contains the six defects shown. If we take linear algorithmic approach to allocate spares, then the allocation shown in the fig.(b) would be the outcome and the repair would not be successful. A successful allocation is possible in this case however as shown in fig.(c).

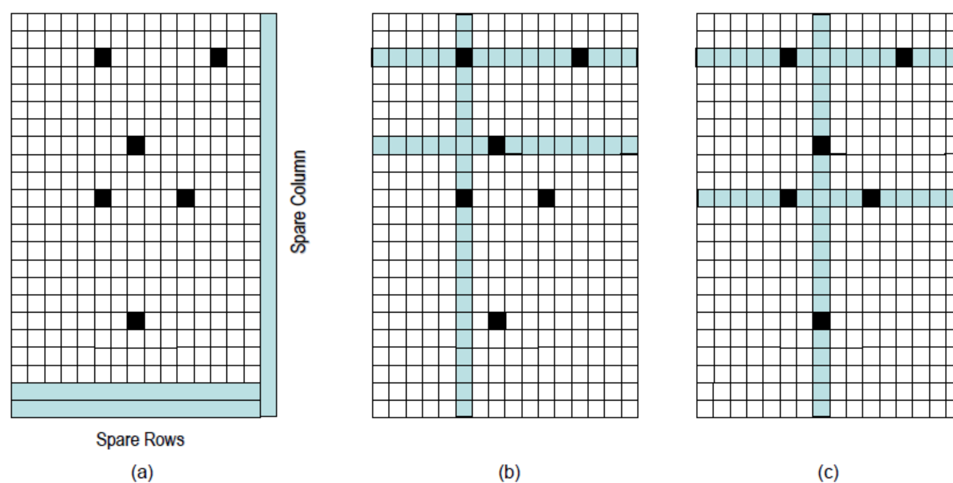


Figure 5.5: Case: Optimal Spare Allocation

If number of rows and columns are relatively small, an optimal solution can be computed.

3. **Repair Delivery:**

   There are two types of repair delivery: Hard repair and Soft repair.

   - **Hard repair:** In this approach, the repair instructions are stored permanently within the die through the programming of fuses. There are two kinds of fuses viz. Laser and Electrical. Laser fuses are programmed by cutting a metal link, while the electrical fuses are either single-time programmable or flash-memory type elements which can be programmed using a certain amount of voltage level. Electrical fuses are smaller in size so their usage is increasing swiftly. Also, they don't require any special equipment or different test insertion to be programmed.

   - **Soft repair:** In this approach, repair instructions are stored into volatile memory, typically in scan registers at each powering up of the device. It has the advantage of storing the repair instructions when each new repair instructions can be stored throughout the life of the device. That gives the disadvantage of creating an extra load onto the circuitry as the data coming from many memories and their devices need to be properly organized while storing it. Henceforth, this approach is exclusively associated with the BIRA mechanism for the calculation of repair instructions on-chip at the power-up.

   - **Self repair:** BISR consists of BIRA and soft repair capabilities. But, a major drawback in this approach is that repair instructions are calculated when the power is up, but the defects occurring under specific conditions (e.g. Higher temperature) won't be taken into the account. So, a combination of both hard and soft repair capabilities are incorporated in advanced BISR solutions. This approach gives advantages like:

     - Simpler test and manufacturing process,
     - Long-term reliability support using soft-repair approach, and
     - Saving in the silicon area by pooling of fuse data.

However, this approach also includes a disadvantage of longer power-up cycle time because it needs BIST to be executed twice. This can be troublesome for some applications.

In BISR architecture, a key component of this architecture is the concept of a centralized fuse pool (eFuse Array) allowing better fuse optimization. On-chip management of this programmable fuse pool is done by a fuse controller. This controller along with one or more BIST controllers perform all necessary activities for testing and repairing the memories. A dedicated chip-wide (BISR) scan chain is used by the fuse controller to transfer fuse data to and from the eFuse array and the various memories. This scan chain contains a BISR register for each memory with redundancy. BISR architecture of LogicVision® is shown below: [25]

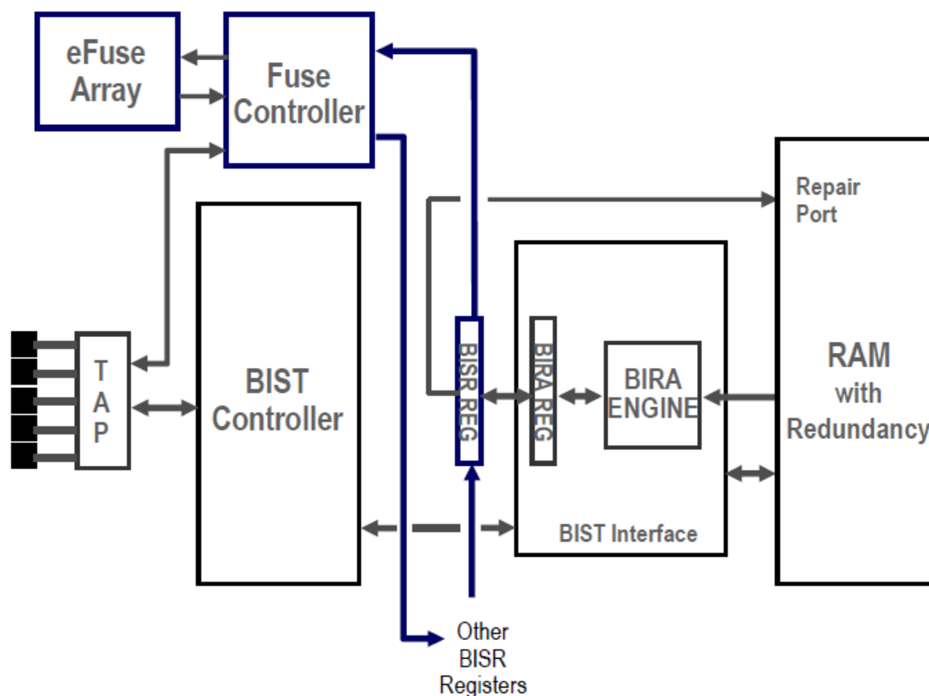


Figure 5.6: BISR Architecture

5.4.1 Advancements in Memory Repairs

Along with the BISR methodology, there are advanced BIST architectures proposed which can overcome the traditional BIST architectures' flaws. Traditional BIST controllers can run test algorithms in a pre-planned sequence during manufacturing test. As these controllers are modified in a definite way, the area estimation is normally low and tests can be applied at-speed.

One of the disadvantages here is their limited flexibility. So, there is a need arisen for certain programmable BIST solution so that some flexibility can be achieved in the test programs at run time. The proposed Field Programmable BIST (FP-BIST)'s top-level architecture is shown in the below figure. [26]

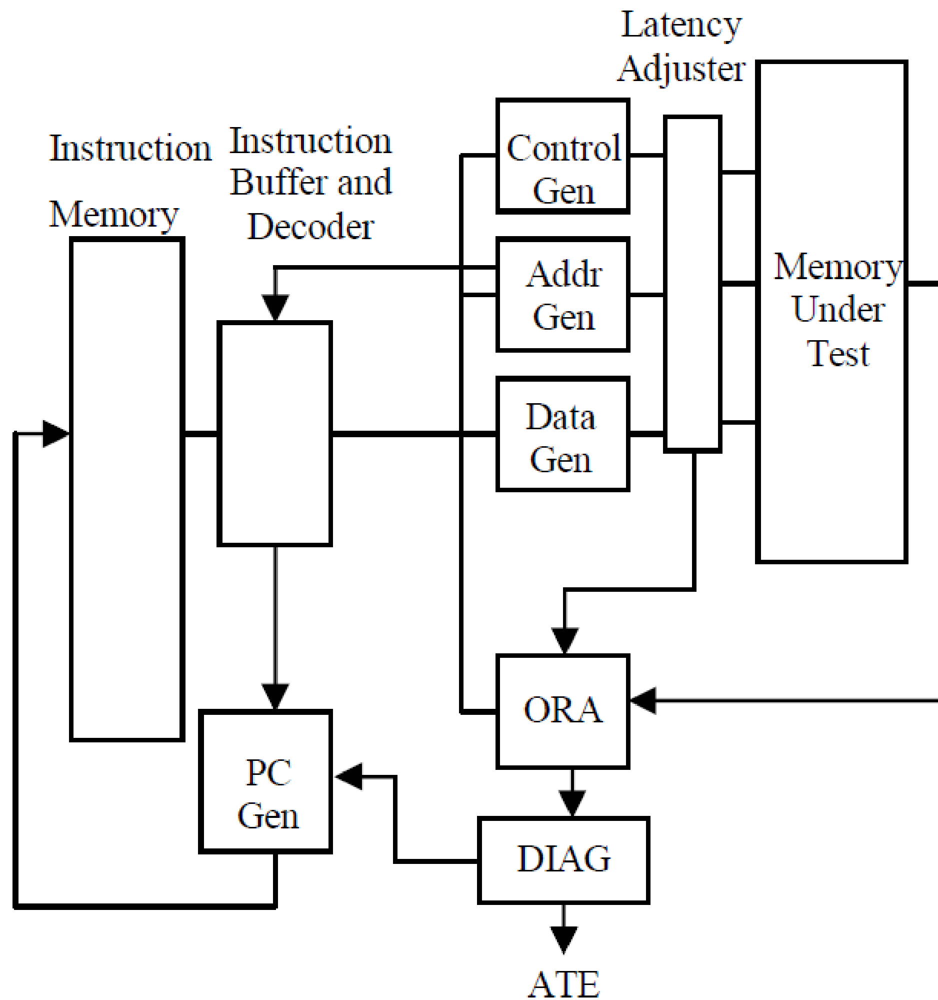


Figure 5.7: FP-BIST Top-level Architecture

Blocks mentioned above:

- *Instruction memory*: The Program counter(PC) chooses the instruction which is to be loaded in the instruction memory. The word size of an instruction is set at 9. Instruction memory consists of registers. The instruction is loaded here from the Automatic Test Equipment (ATE) into the instruction buffers via multiplexers.

- *Instruction decoder*: It decodes and reads configuration from the instruction memory, and reads algorithm instructions from the instruction buffers.

- *Pc generator*: It generates three addresses for accessing the instruction memory, base and local loop buffers.
- *Control signal generator*: It generates read/write signal to the memory under test.
- *Address generator*: It generates valid address sequences to the memory under test.These sequences depend on different addressing schemes specified in the configuration instructions.
- *Data generator*: It generates test stimulus written to the memory array and generates desired data read from the same.
- *Latency adjuster*: It synchronizes the control signals, address and data generated in different clock cycles. Since extensive pipelining is supported for operating the controller at very high speed, the adjuster also controls the discontinuity of the memory input data so that the memory output data is compared with the right and desired data.
- *Output Response Analyzer*: It evaluates the memory output. It compares the expected data and actual memory output read from different address locations. If the number of bits in a word is large (i.e. in the case of very broad memories), pipelining stages are inserted to operate the comparator at full-speed.
- *Diagnostic monitor*: It captures and scans out diagnostic data from the controller to the ATE.

Some faults are obvious as these memories are integrated on the SoC. As discussed in [27], some of the faults are Stuck-at Faults (SAFs), Transition Faults (TFs), Coupling Faults (CFs), Address Decoding Faults (ADFs) and Physical neighborhood pattern-sensitive faults (NPSFs). This commonly seen faults can be resolved by MBIST as discussed in Section-5.2. For testing of core as well as noncore elements in many/multi-core processors, latest researchs have proposed two online self-test methodologies which are 1.Concurrent Autonomous Self-Test using Stored Patterns (CASP) and 2.Virtualization-Assisted Concurrent Autonomous Self Test (VAST). The CASP [28] and VAST [29] algorithms supports concurrent self-testing of SoCs having both core components and non-core components by way of 1) Resource Allocation and Sharing techniques, 2) No-performance-impact testing, and 3) Smart backups.

5.5 Automation Introduced

Pre-requisites:

- **Scripting Language chosen:** Python. Tcl can also be used.
- **Commands:** Pflow commands related documentation.
- The user needs to be in the respective project/scratch area where this operation can be performed.
- Creation and working in for such scratch areas needs to be done with the help of an expert in order to have a smooth operation.

Execution:

As mentioned in the Section 5.3.1, a simple python script was developed for executing the given Pflow commands. Pflow process contains total 04 commands which need to be executed sequentially in order to generate MBIST wrappers. The script was run on the respective project area. A trial run was conducted on MS-Visual Studio-2017 Enterprise Edition™. Screenshot for the script output is as shown below:

```
Enter your .xls file:    A
Enter your config file:  B
Enter your Out dir name: C
Enter your Unit name:D
pflow setup -memlist A -soc_cfg B -subsyslist all -outdir C -ftool ts -unit D -source_location MemoryListTools
pflow run -outdir C -unit D -source_location MemoryListTools
setenv SYNOPSYS_SIM_SETUP Path to synopsys_sim.setup
Unit pathC/memory_wrapper/filelist_v.list
pflow sim -outdir C -unit D -v_f Unit pathC/memory_wrapper/filelist_v.list
```

Figure 5.8: Automation Script Output

Here, the user inputs are given as dummy values and explained below:

1. **A:** An .xls file which contains all the memory specifications.
2. **B**: A .json configuration file containing further specifications.
3. **C**: Desired output directory name, and
4. **D**: Working Unit name.

Upon giving the above details, all four commands get executed and the mbist wrappers get generated into the desired directory which can be verified by going into that respective folder.

5.6 Conclusion

- Memory library generation is an important task in realizing the SoC as it plays a vital role in the RF modem attached to the client SoC.
- The memory views compiled after running the libgen script need to be locally checked. A compile run followed by the sanity run was also performed before integrating the views in the main line.
- This sanity run gives errors if any specification mismatch is present in the compiled views. E.g. there are Virage models to be compiled along with the memory views. Tweaking into the compiler script could give a clean compile.
- The automation script ran on the project platform too. However, the same steps couldn't be replicated for the memory library generation as the commands used in it are system commands and thus needs step-by-step execution.
- As mentioned earlier, Tcl can also be used, but it will be a time-consuming effort as every time the output directory name needs to be changed in the Tcl script while sourcing the Tcl file for its execution. Python gives that freedom in generalizing the command execution.

5.7 Possible recommendations

- The python script only runs the list of commands, but the source of the commands, i.e. compilers needs to be identified and changed accordingly.
- Memory compiler version is also an important factor while generating the lib. It affects the integration so choice of the compiler version is to be done carefully.

Chapter 6

Appendix

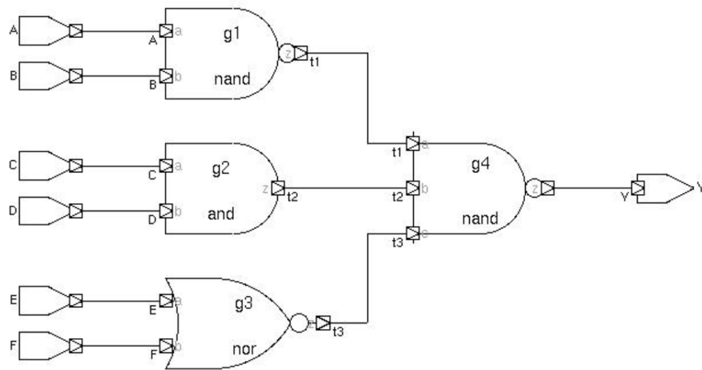


Figure 6.1: Schematic for Verilog example

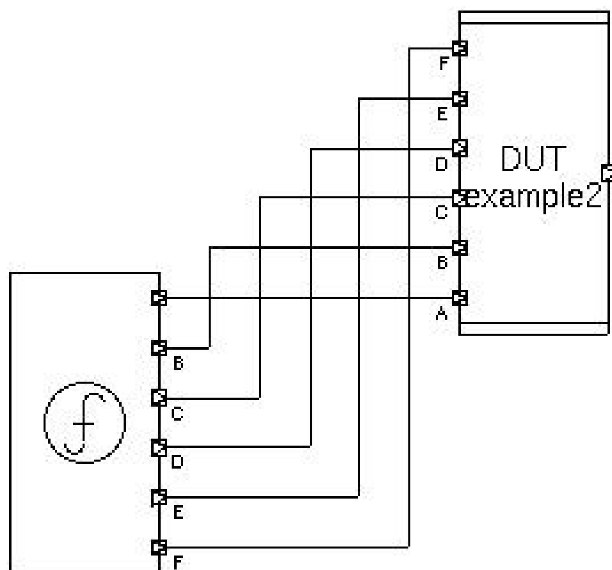


Figure 6.2: Chip realization for Verilog example

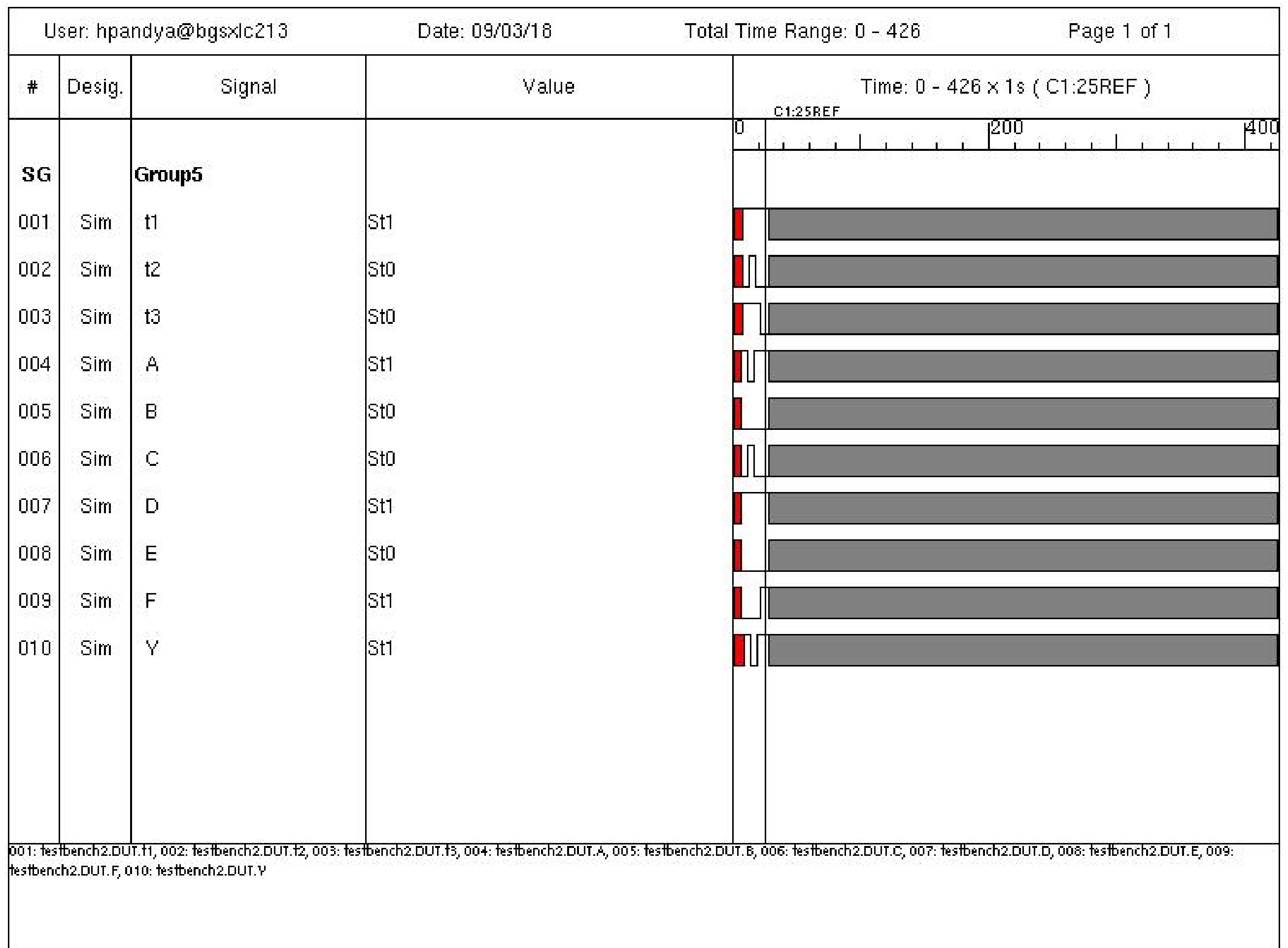


Figure 6.3: Simulation by DVE Tool

Bibliography

[1] Intel, "Support." https://www.intel.com/content/www/us/en/support/articles/000015119/programs.html, July 2018.

[2] "Intel at Mobile World Congress." https://www.intel.in/content/www/in/en/events/corporate/mobile-world-congress.html.

[3] "Intel Mobile Communications Profile." https://www.intel.com/content/www/us/en/wireless-products/mobile-communications/company-overview.html, July 2018.

[4] Intel News Room, "The 5G Revolution-5G will Transform Business and the World." https://newsroom.intel.com/wp-content/uploads/sites/11/2018/09/5G-factsheet.pdf, November 2018.

[5] Intel News Room, "Intel announces world's first global 5g modem." https://newsroom.intel.com/newsroom/wp-content/uploads/sites/11/2017/01/5G-modem-fact-sheet.pdf, July 2018.

[6] Intel News Room, "Intel to Exit 5G Smartphone Modem Business, Focus 5G Efforts on Network Infrastructure and Other Data-Centric Opportunities." https://newsroom.intel.com/news-releases/intel-modem-statement/#gs.ah6tu3, 16 April, 2019.

[7] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*, ch. 9. Willey-IEEE, first ed., 1990. [Online; accessed 25-Oct-2018].

[8] Imperial College, London-Dept. of Computing, "Basic system-on-chip model." https://www.doc.ic.ac.uk/~wl/teachlocal/cuscomp/notes/cc11.pdf, 12 Feb 2019. [Online; accessed 23-Apr-2019].

[9] B. Dubrov, H. Eran, A. Freund, E. F. Mark, and S. Ramji, *Principles and Practice of Constraint Programming-Pin Assignment using Stochastic Local Search Constraint Programming*. September 2013.

[10] D. Sharvil, "ASIC Design Flow Outline." https://asic4u.wordpress.com/, 6th January 2018. [Online; accessed 23-Apr-2019].

[11] T. W. Williams and R. Kapur, "Design for Testability in Nanometer Technologies; Searching for Quality."

[12] Wang, Laung-Terng, Stroud, Charles E, Touba, and Nur A, *System-on-Chip Test Architectures*.

[13] M. A. . M. A. B. . A. D. Friedman, *Digital Systems Testing and Testable Design*, ch. 9, pp. 343–419. Wiley-IEEE Press, 1 ed., 1990.

[14] H. s. Fung and S. Hirschhorn, "An automatic dft system for the silc silicon compiler," *IEEE Design Test of Computers*, vol. 3, pp. 45–57, Feb 1986.

[15] T.-H. Chen and M. A. Breuer, "Automatic design for testability via testability measures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 4, no. 1, pp. 3–11, 1985.

[16] H. Fung and S. Hirschhorn, "An automatic DFT system for the SILC silicon compiler," *IEEE Design & Test of Computers*, vol. 3, no. 1, pp. 45–57, 1986.

[17] E. J. Marinissen, B. Prince, D. Keitel-schulz, and Y. Zorian, "Challenges in Embedded Memory Design and Test," pp. 722–727, March 2007.

[18] I. Kim, Y. Zorian, G. Komoriya, H. Pham, F. P. Higgins, and J. L. Lewandowski, "Built in self repair for embedded high density SRAM," in *Proceedings International Test Conference 1998 (IEEE Cat. No.98CH36270)*, pp. 1112–1119, Oct 1998.

[19] S. Shoukourian, V. Vardanian, and Y. Zorian, "SoC Yield Optimization via an Embedded-Memory Test and Repair Infrastructure," *IEEE Design Test of Computers*, vol. 21, pp. 200–207, May 2004.

[20] A. C. Cheng, "Comprehensive Study on Designing Memory BIST: Algorithms, Implementations and Trade-offs," tech. rep., Department of Electrical Engineering and Computer Science,The University of Michigan, Michigan, 2002.

[21] A. Kaushal and K. Kathuria, "MBIST verification: Best practices & challenges," 2014.

[22] M. Miyazaki, T. Yoneda, and H. Fujiwara, "A Memory Grouping Method for Sharing Memory BIST logic," 2006.

[23] P. Buntoro David and N. Desilva, "Intel Tessant MBIST and Parallel Flow Documentation."

[24] Intel Documentation-App Note, "Generation of third-party Scan-Insertion tool commands template," July 2016.

[25] Mentor Graphics Documentation, "Memory Repair Primer–A Guide to Understanding Embedded Memory Repair Options and Issues," November 2008.

[26] X. Du, N. Mukherjee, C. Hill, W. Cheng, and S. Reddy, "A Field Programmable Memory BIST Architecture Supporting Algorithms with Multiple Nested Loops," in *2006 15th Asian Test Symposium*, pp. 287–292, Nov 2006.

[27] G. P. Acharya and M. A. Rani, "Survey of test strategies for system-on chip and it's embedded memories," in *2013 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*, pp. 199–204, Dec 2013.

[28] Y. Li, S. Makar, and S. Mitra, "CASP: Concurrent Autonomous Chip Self-Test Using Stored Test Patterns," in *2008 Design, Automation and Test in Europe*, pp. 885–890, March 2008.

[29] H. Inoue, Y. Li, and S. Mitra, "VAST: Virtualization-Assisted Concurrent Autonomous Self-Test," in *2008 IEEE International Test Conference*, pp. 1–10, Oct 2008.