"STUDY AND IMPLEMENTATION OF BIST FOR 65NM HIGH SPEED MEMORY"

Major Project Report

Submitted in Partial Fulfillment of the Requirements for Semester III-IV
of

MASTER OF TECHNOLOGY
IN
ELECTRONICS & COMMUNICATION ENGINEERING
(VLSI Design)

By
Keerti Choubey
04MEC005

Department of Electronics & Communication Engineering
INSTITUTE OF TECHNOLOGY
NIRMA UNIVERSITY OF SCIENCE & TECHNOLOGY,
AHMEDABAD 382 481

May 2006

CERTIFICATE

This is to certify that the M.Tech. Dissertation Report entitled **"Study and Implementation of BIST for high Speed 65nm Memory"** submitted by **Ms. Keerti Choubey (Roll no. 04MEC005)** towards the partial fulfillment of the requirements for Semester III-IV of **Master of Technology (Electronics and Communication Engineering)** in the field of **VLSI Design** of **Nirma University of Science and Technology, Ahmedabad** at **S.T. Microelectronics, Noida** is the record of the work carried out by her under our supervision and guidance. The work submitted has in our opinion reached a level required for being accepted for examination. The results embodied in this Dissertation-Project work to the best of our knowledge have not been submitted to any other University or Institute for award of any degree or diploma.

Date:

Project Guide
Mr. Naveen Shrivastava
ST Microelectronics
Noida (U.P.)

Internal Project Guide
Prof. Mrs. Usha S. Mehta
Institute of Technology
Nirma University,
Ahmedabad (Gujaraat)

Course Co-ordinator
Dr. N.M. Devashrayee
VLSI Design,
Institute of Technology,
Nirma University, Ahmedabad

HOD
Dr. M.D. Desai
EE – Department
Nirma University, Ahmedabad

Director
Dr. H.V. Trivedi
Institute of Technology,
Nirma University, Ahmedabad

ACKNOWLEDGEMENT

Any fruitful effort in a new work needs a direction and guiding hands that shows the way. It is my proud privilege and pleasure to bring indebtedness and warm gratitude to respect Mr. Ameet Mattoo, Section Manager, Modeling, Soft IP and Test Solution (MST) group, ST Microelectronics, Noida, for giving me an opportunity to carry out my project work with MST group.

I would like to express my profound gratitude to my project guide Mr. Naveen Shrivastava, Senior Design Engineer, Test Solution group, STMicroelectronics, Noida, for his outstanding support and guidance during my project work.

I am thankful to my team members Mr. Prashant Dubey, Mr. Akhil Garg, Mr. Nitin Sharma, Mr. Shashi Bhusan Garg, Mr. Sravan Bhaskrani and Mr. D. R. Madhav who have obliged me with their time-to-time guidance.

I would like to express my warm thanks to my internal project guide Prof. Mrs. Usha S. Mehta for her kind support and guidance during my project work.

Also, I would like to express my warm gratitude to Dr. N. M. Devashrayee, Course Coordinator, who has always been a constant source of encouragement and support.

I am delighted to express my warm thanks to all my colleagues and friends, especially, Mr. Jatin Fultaria, Mr. Apurva Chaure, Mr. Pankaj Agrawal, Ms. Tripti Bhargava and Ms. Payal Kapadia, for their continuous support and constant encouragement during project work. I would also like to thank each and everybody who has directly or indirectly helped me in the accomplishment of the project.

I am grateful to my parents and all other family members for their understanding, love and endless support, which make my work possible.

And at last I am extremely thankful to **ST Microelectronics, NOIDA** for providing me with an opportunity to work with them and undertake a project of such importance.

(Keerti Choubey)

ABSTRACT

In today's era of System on Chip (SoC) a custom chip is composed of different embedded modules such as microprocessor, analog and mixed signal logic, digital logic and of course the integral part of all i.e. memories. More than 50% of all the designs today have embedded memories implemented in them which cover up more than 60% of total die area. While embedded memory presents significant system performance and cost reduction advantages, it also brings its own testing issues. Test vector style are not suitable for verifying embedded memory arrays, as they are too costly because of the time spent in the manufacturing tester grows exponentially as the embedded memory die area increases. This problem can be alleviated by implementing embedded memory built in self-test (BIST). In simplistic terms Memory BIST is an on-chip utility that enables the execution of a proven set of algorithmic style verification tests directly on the memory array. These tests can be executed at the design's full operating frequency to prove the memory array operations and identify errors caused by chip defects.

This thesis work consists of study and development of Memory BIST for memories of different size and types. Analysis and modification work is done on new shared BIST architecture for multiple memories. The analysis has proved that the new BIST architecture greatly improves the speed performance of the BIST while reducing the area. Study of BIST compiler development through C programming and shell scripting in UNIX environment has been done to support the team for compiler development. Major development work has been carried out on dedicated BIST for 65nm high speed Single-Port SRAM memory.

After the insertion of BIST into the design, validation of new design is done which includes simulation, synthesis, gate simulation, formal equivalence check between different levels of RTL, and code coverage analysis. For the automation of these validation tasks a tool has been made using the shell programming in UNIX environment. The updation of the shell script of this tool is a part of this project work.

COMPANY PROFILE

Company	ST MICROELECTRONICS Pvt. Ltd.
Head Office	web site: http://www.st.com
Contact Phone	91 120 2512021-30 91 120 2515262-64
Incorporated in	June 1987
CEO	Carlo Bozotti
Address	Plot No.1, Knowledge Park III Greater Noida – 201308
No. of Employees	Approximately 50,000.
Branches in India	Bangalore, Greater Noida

About The Company:

ST Microelectronics is a global independent semiconductor company and is a leader in developing and delivering semiconductor solutions across the spectrum of microelectronics applications. ST is one of the world's largest semiconductor companies, with net revenues of US$8.88 billion in 2005.

According to the latest industry data, ST is the world's fifth largest semiconductor company and has leading positions in sales of Analog Products, Application Specific Integrated Circuits ("ASICs") and Application Specific Standard Products ("ASSPs"). ST is also number one in camera modules, number two in discrete and analog, and number three in NOR Flash, as well as in the application segments of Automotive, Industrial, and Wireless. ST is also a leading supplier of semiconductors for set-top boxes, smart cards, and power management devices.

Company's Clients:

The Company currently offers over 3,000 main types of products to more than 1,500 customers worldwide.

Product Range:

ST Microelectronics produces a diverse range of semiconductors - from single transistors to microprocessors with millions of components on the same 'silicon chip' - which can be found in many disparate products or environments - from high performance supercomputers to everyday items such as telephones, cars, toasters, or even light bulbs.

Dedicated ICs

These are integrated circuits that are designed for use in a specific application. Examples include modem chips; digital set top box chips, motor control circuits, audio amplifiers and even such highly specialized circuits as fluorescent lamp ballasts.

One of the most important application areas for dedicated ICs is the rapidly expanding market for digital consumer products. These include multimedia PCs, set top boxes (STB), and Digital Video Disc (DVD) players. At the heart of most of these products is the digital image compression technology known as MPEG. ST is the world's number one manufacturer of MPEG chips according to independent industry analysts Dataquest.

Microprocessors and Other Programmable Devices

ST offers a comprehensive range of microcontrollers, microprocessors and programmable devices. These range from robust 8-bit Flash Microcontrollers that cost much less than a dollar in volume through 16 and 32-bit Flash Microcontrollers to 32-bit RISC and CISC microprocessors, multimedia accelerators, and complex SOCs embedding several micro and DSP core architectures.

Semi custom Products

Semi custom devices are integrated circuits designed with the help of cell libraries and powerful Computer-Aided Design (CAD) software. This technique allows devices that meet a customer's specific requirements to be developed quickly and cost-effectively by combining building blocks from libraries of tried and tested circuits.

Discrete Memories

ST's main focus in the memory market is on non-volatile memories, an area that includes EPROM, EEPROM and flash memories. The Company is the world's number one supplier of non-volatile EPROM memories and is one of the top five manufacturers of flash memory. Other memory products include specialty SRAMs and ZEROPOWER battery backed memories.

Standard ICs

ST offers a broad range of devices such as op amps, comparators, voltage regulators and logic functions such as NAND gates and inverters, which perform standard functions that

are used in all types of applications. In recent years, the Company has achieved great success in the standard linear market, raising its market share from 3.1% in 1991 to 6.9% in 1995 (World Semiconductor Trade Statistics Ffures). Results for the same period from another independent source, Dataquest, also show a rise in world ranking from No. 13 (1991) to No. 4 (1995).

Discrete

Discrete devices can be made to handle greater power levels and higher voltages than integrated circuits and their main use is to act as power switches and power amplifiers. ST is one of the world's leading suppliers of discrete power devices, with a product range that includes MOSFETs, bipolar transistors, IGBTs and triacs.

Table of Contents

PROJECT TITLE (i)

CERTIFICATE (ii)

ACKNOWLEDGEMENT (iii)

ABSTRACT (iv)

COMPANY PROFILE (v)

LIST OF FIGURES (xii)

LIST OF TABLES (xiii)

1. CHAPTER 1: INTRODUCTION 1

1.1 Digital Test Methodologies: ATE vs. BIST 3

1.2 System-on-a-Chip Test Challenges 4

1.2.1 Motivation For A Shift From ATE-Based SOC Testing To BIST 5

1.3 Embedded Memory Testing 9

1.4 Thesis Organization 10

2. CHAPTER 2: Memory Testing Methodologies 12

2.1 Memory Concepts 12

2.1.1 RAM Basics 13

2.1.2 Functional RAM Chip Model 14

2.1.2.1 Basic Description Of The Blocks 15

2.2 Memory Modeling 15

2.2.1 Address And Data Scrambling And Descrambling 16

2.2.1.1 Address Scrambling 17

2.2.1.2 Address Descrambling 18

2.2.1.3 Data Scrambling 18

2.2.1.4 Data Descrambling 19
2.3 Fault Modeling 21
2.3.1 Stuck-at Faults 23
2.3.2 Transition Faults 23
2.3.3 Coupling Faults 23
2.3.4 Neighborhood Pattern Sensitive Faults 25
2.3.5 Address Decoder Faults 25
2.4 Functional Testing and March Test Algorithms 26
2.4.1 Characteristics of March Algorithms 27
2.4.2 March Algorithms with Diagnosis and Repair Support 28
3. CHAPTER 3: Memory BIST Methodologies 29
3.1 Memory BIST Approaches 29
3.2 Memory BIST Architectures 32
3.2.1 Dedicated BIST Methodology 33
3.2.1.1 Design Implementation 37
3.2.2 Distributed MBIST Architecture 39
4. CHAPTER 4: BIST for 65nm High Speed Single-Port Memory 41
4.1 Scrambling 41
4.2 Redundancy 42
4.3 Fault Models of Single-port High Speed SRAM 42
4.4 Algorithm 43
4.4.1 mMarchLR 14N 43
4.4.2 Mask Bits Test 47
4.4.3 Data Bit Coupling Test 48
4.4.4 CSN Bit Coupling Test 49
4.4.5 Address Delay Decoder Test 49
4.4.5.1 Fully Decoded Memory 50
4.4.5.2 Not Fully Decoded Memory 51
4.5 Implementation 52

4.5.1 I/O Pin Description 52
4.5.2 Full Pin Description 53
4.5.3 Operating Mode Description 54
4.5.3.1 Scan ATPG Mode 54
4.5.3.2 Transparent Mode 55
4.5.3.3 Run BIST for RAMs 55
4.5.3.4 IDDQ Fill 1/Fill 0 Modes and Retention Test 56
4.5.3.5 Scan Collar Mode 58
4.5.3.6 BITMAP Mode 60
5. CHPATER 5: VALIDATIONWIZARD & RESULTS 63
5.1 Validation 63
5.2 ValidationWizard 64
5.2.1 Running the ValidationWizard 66
5.3 The Shell Scripts 66
5.4 Validation Of BIST for High Speed Single-port SRAM 67
5.4.1 Simulation and Gate Simulation 67
5.4.2 Synthesis 68
4.4.3 Formal Verification 68
4.4.4 Test Coverage 68
4.4.5 HAL Run 68
5.5 RESULTS 68
6. CHPATER 6: CONCLUSION & FUTURE SCOPE OF WORK 69
7. REFERENCES 70

List of Figures

Sr. No.	Title	Page No.
1.1	VLSI circuit transistor density	2
1.2	Basic Principal of Digital Testing	3
2.1	Functional RAM chip model	14
2.2	Address descrambled block inserted in the test path	18
2.3	Data Descrambling Example	19
2.4	Data Descrambler Insertion	20
2.5	Address Decoder Faults	26
2.6	Combination of Address Decoder Faults	26
3.1	Generic BIST Architecture for Standalone memory	35
3.2	Elaborated BIST Execution diagram	38
3.3	Generic Shared MBIST Architecture	39
4.1	Pin connections (BIST AND MEMORY)	52
4.2	Scan ATPG Flow	54
4.3	Run BIST Execution	55
4.4	Fill 0 and Fill 1 operation	56
4.5	Fill 0/1 and Retention test flow	57
4.6	Scan Collar Implementation	58
4.7	Scan Collar Mode test Flow	59
4.8	BITMAP Mode Test Flow	61

List of Tables

Sr. No.	Title	Page No.
2.1	Functional Memory Faults	21
2.2	Reduced Functional Memory Faults	22
2.3	Relationship Between Functional and Reduced Functional Faults	22
4.1	Modified MarchLR algorithm	43
4.2	Mask Bits Algorithm	47
4.3	Mask Bits Algorithm (continued..)	47
4.4	Data bits coupling algorithm	48
4.5	CSN bit Coupling test	49
5.1	The Tools used for different validation tasks	64

CHAPTER 1

INTRODUCTION

A VLSI chip is manufactured through a series of steps that involve chemical, metallurgical, and optical processes. Out of a set of chips generated, if the yield of good chip is 75% then on an average 25% of the manufactured chips will be faulty. Thus, at the end of VLSI manufacturing process we always have “testing,” which isolates the good chips from bad ones. Manufacturing test helps to detect physical defects (e.g., shorts or opens) prior to delivering the packaged circuits to end-users. Inadequate testing will have some faulty chips shipped to the customer .At the same time, the cost of testing directly increases the over all cost of the chip.

Due to the rapid progress in the very large-scale integrated (VLSI) technology, an increasing number of transistors can be fabricated onto a single silicon die. Transistor feature size on a VLSI chip reduces roughly by 10.5% every year, resulting in a transistor density increase of roughly 22.1% every year. For example, a state-of-the-art 130 nm complementary metal-oxide semiconductor (CMOS) process technology can have up to eight metal layers, poly gate lengths as small as 80 nm and silicon densities of 200K-300K gates/mm2 [10]. The on-chip clock frequency has also increased up to 1 GHz for the latest VLSI circuits. However, although milliongates integrated circuits (ICs) can be manufactured, the increased chip complexity requires robust and sophisticated test methods. Hence, manufacturing test is becoming an enabling technology that can improve the declining manufacturing yield, as well as control the production cost, which is on the rise due to the escalating volume of test data and testing times. Therefore reducing the cost of manufacturing test, while improving the test quality required to

achieve higher product reliability and manufacturing yield, has already been established as a key task in VLSI design.

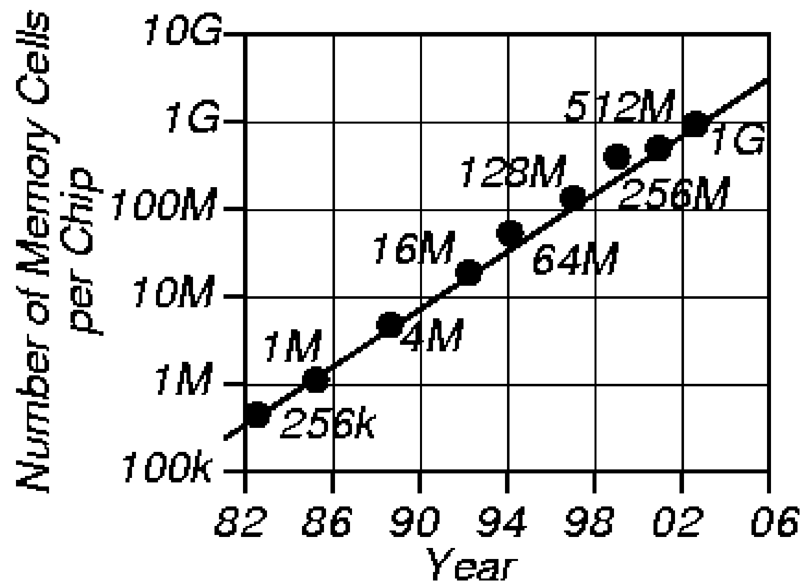


Figure 1.1: VLSI circuit transistor density [1]

For a VLSI chip to be manufactured, we must have a verified design and a set of tests. The following questions characterize testing of complex systems-

- Can tests that detect all faults be assured?
- Can test development time be kept low enough to be economical?
- Can test execution time be kept low enough to be economical?

Design for testability (DFT) refers to those design practices that allow us to answer the above questions in the affirmative. There are specific DFT techniques for each type of component in an electronic system.

1.1 Digital Test Methodologies: ATE vs. BIST

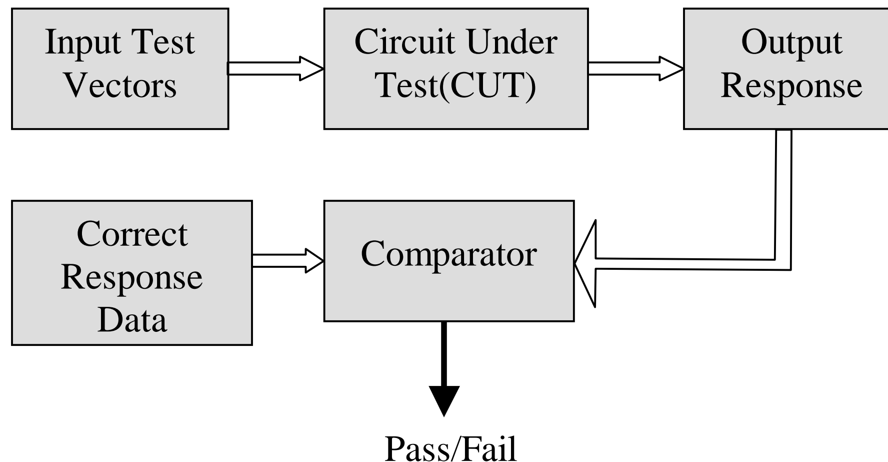


Figure 1.2: Basic Principal of Digital Testing

The basic principle of manufacturing testing is illustrated in Figure 1.2. Circuit under test (CUT) can be the entire chip or only a part of the chip (e.g., a memory core or a logic block). Based on the techniques how the test vectors are applied to the CUT and how the output responses are compared, there are two main directions to test electronic circuits: external testing using automatic test equipment (ATE) and internal testing using built-in self-test (BIST). When external testing is employed, the input test vectors and correct response data are stored in the ATE memory. Input test vectors are generated using ATPG tools, while correct response data is obtained through circuit simulation. For external testing, the comparison is carried out on the tester. Although the ATE-based test methodology has been dominant in the past, as transistor to pin ratio and circuit operating frequencies continue to increase, there is a growing gap between the ATE capabilities and circuit test requirements. The main limitations of ATE are:

- The time spent in the manufacturing the tester grows exponentially as the embedded memory die area increases.

- The up gradation of test equipment is not happening with the same rate due the high cost involved.
- Design operating frequencies have increased up to 2 GHz.
- I/O pads fail at high frequencies (~ 400 MHz).
- Extra interconnects increase the routing congestion.

ATE limitations make BIST technology an attractive alternative to external test for complex chips. Embedded RAM memories are perhaps the hardest type of digital circuit to test, because memory testing requires delivery of a huge number of pattern stimuli to the memory and the readout of an enormous amount of cell information. The difficulty and time required to propagate all of that information through the various glue logic and busses in an embedded core chip almost forces the use of memory BIST.

BIST is a design-for-test (DFT) method where part of the circuit is used to test the circuit itself (i.e., test vectors are generated and test responses are analyzed on-chip). BIST needs only an inexpensive tester to initialize BIST circuitry and inspect the final results (pass/fail and status bits). However, BIST introduces extra logic, which may induce excessive power in the test mode, in addition to potential performance penalty and area overhead.

It is important to note that the main problem with logic BIST lies in the computational overhead required to synthesize compact and scalable test pattern generators and response analyzers such that high fault coverage is achieved in low testing time and with limited interaction to external equipment. In contrast, due to the regular memory block structure and simple operations of memory cores, memory BIST (MBIST) can be implemented using compact and scalable test pattern generators and response analyzers and it can rapidly achieve high fault coverage for certain functional fault models

1.2 System-on-a-Chip Test Challenges

As process technologies continue to shrink, designers are able to integrate all or most

of the functional components found in a traditional system-on-a-board (SOB) onto a single silicon die, called system-on-a-chip (SOC) . This is achieved by incorporating pre-designed components, known as intellectual property (IP) cores (e.g., processors, memories), into a single chip. While SOCs benefit designers in many aspects, their heterogeneous nature presents unique technical challenges to achieve high quality test, i.e., acceptable fault coverages for the targeted fault models.

1.2.1 Motivation For A Shift From ATE-Based SOC Testing To BIST

- **Controllability and Observability:**

  Since most of the input/output (I/O) ports of these embedded cores are not directly connected to the SOC's pins, the testability , i.e., both the controllability and the observability [1], is reduced and, unless some special DFT techniques are employed, the fault coverage will be lowered.

  However, when ATE-based testing is employed (i.e., patterns and responses are stored on the tester), since the number of tester channels is limited in practice, test concurrency is bounded by the number of these channels, which can adversely influence the cost of test. This problem can be addressed by moving the generation and analysis functions on-chip and use an inexpensive tester to initialize, control and observe the final results of the testing process.

- **Volume of test data, tester channel capacity and testing time**

  The volume of test data is determined by the chip complexity and it grows rapidly as more IP cores are integrated into a single SOC. The easiest way to deal with increased volume of test data is to upgrade the tester memory and use more tester channels to increase test concurrency, however this is infeasible since it will prohibitively increase the ATE cost. A more cost effective approach is to use test data compaction and/or compression. Test data compaction reduces the number of test patterns in the test set (by discarding test patterns that target faults detected by

other patterns in the test set) and test data compression decreases the number of bits (that need to be stored for each pattern) and uses dedicated decompression hardware (either off or on-chip) for real-time decompression and application [9]. Test data compaction reduces the volume of test data, however it is trading-off the tester channel capacity against the testing time. If the decompression hardware is placed on-chip, then test data compression eliminates this trade-off. Deterministic BIST is a particular case of test data compression where the compressed bits are used for BIST initialization (i.e., seeds) and BIST observation (i.e., signatures). The benefits of memory BIST technology are justified mainly by its deterministic nature.

- **Heterogeneous IP cores**

  Many SOC designs incorporate cores that use different technologies, such as random logic, memory blocks, and analog circuits. For SOC testing one can use generic high-performance mixed-signal ATEs, however their high production cost brings limited benefits to complex designs, since cores using heterogeneous technologies still need to be tested sequentially, thus lengthening the testing time and ultimately raising the manufacturing test cost.

  In addition, embedded core controllability and observability issues cannot be addressed without dedicated on-chip DFT hardware, whose necessity justifies a shift toward BIST. The use of different BIST circuitry for the appropriate technologies (logic, memory or analog BIST), increases both testability and test concurrency of SOCs comprising heterogeneous IP cores.

- **At-speed test**

  As VLSI technology moves below 100 nm, traditional stuck-at fault testing is not sufficient. This is because unanticipated process variations, weak bridging defects, and crosstalk violations (only to mention a few) may cause only timing malfunctions, which cannot be detected by the stuck-at fault test vectors delivered

by ATEs whose frequency is lower than the maximum CUT frequency. These logical faults caused by timing-related defects are known as delay faults and they can only be detected when the chip is tested at the functional (rated) speed. This type of test is called at-speed test. at-speed test can be performed using high-speed ATEs (however, even the highest performance/cost ATEs will be slower than the fastest new chips), or more cost effectively, by BIST interacting with a low-speed testers required only to activate the self-test circuitry and to acquire the BIST signatures.

- **Power dissipation**

  Power dissipation is becoming a key challenge for the deep sub-micron CMOS digital integrated circuits. Placing more and more functions on a silicon die has resulted in higher power/heat densities, which imposes stringent constraints on packaging and thermal management in order to preserve performance and reliability [11]. There are two major sources of power consumption in CMOS VLSI circuits: dynamic power dissipation, due to capacitive switching, and static power dissipation, due to leakage and subthreshold currents. The 2001 International Technology Roadmap for Semiconductors (ITRS) [12] anticipates that power will be limited more by system level cooling and test constraints than packaging. This is because, if packaging and thermal management parameters (e.g., heat sinks) are determined only based on the functional operating conditions, the higher test switching activity [12] and test concurrency will affect both manufacturing yield and reliability [11].

  On the one hand, dynamic power dissipation dominates the chip power consumption for digital CMOS technology in 180 nm range or higher. Dynamic power dissipation can be analyzed from two different perspectives. Average power dissipation which stands for the average power utilized over a long period of operation, and peak power dissipation which is the power required in a very short time period such as the power consumed immediately after the rising or

falling edge of the system clock. When considering SOC test, to achieve high fault coverage with less test data, the test patterns are usually uncorrelated [2]. This means the switching activity during test can differ from that during functional operation. In most cases, the testing power consumption is the higher one. The switching activity is 35-40% more during scan-based transition test than that in normal functional mode. For traditional stuck-at fault test, one straightforward solution to meet the power constraints is to reduce the system clock frequency during test which implies longer testing time. However, as descry –bed in the previous challenge, to test time related faults, at-speed testing is necessary. Consequently, the power dissipation during at-speed test can exceed the maximum power limit which may lead to chip malfunctions or to burn the overheated chip.

On the other hand, static power dissipation is becoming an important component for low power design and test in 130nm or lower CMOS technologies with low gate subthreshold. Power gating is an efficient method to reduce static power dissipation and it based on disconnecting the idle module(s) from the power and ground network to reduce the leakage currents. This technique is particularly useful for SOCs with a high number of embedded memories.

All the above mentioned SOC test challenges need to be overcome in order to reduce the ever-growing cost of manufacturing test while enabling high manufacturing yield and reliability through satisfactory test quality. Although the cost of test is dominated by many factors, such as the cost of production ATEs, testing time, performance of test automation tools (e.g., ATPG), area and performance overhead caused by additional DFT or BIST circuitry, it is essential to balance this cost against the benefits of enabling high product reliability and a fast yield learning curve. As the SOC complexity increases and more physical defects manifest themselves only in the timing domain, at-speed BIST is emerging as an essential and necessary technology, which can enable short time-to-volume and low cost of manufacturing test. This is also correlated to the fact

that, as total chip area continues to increase, the overhead associated with consciously designed BIST architectures is decreasing. The focus of this thesis is to development of cost-effective BIST architectures for embedded memory testing.

1.3 Embedded Memory Testing

Memory cells are designed using transistors and/or capacitors, and therefore logic gates cannot model them. Structural test based on gate level netlist cannot be applied to memory testing. However, as mentioned in the previous section, memory cores have a rather regular structure caused by identical memory cells and very simple functional operations (only read and write), which are very suitable for functional test. Unlike the case of random logic testing, which needs a large set of deterministic test patterns to reach the desired fault coverage, functional test programs for embedded memory cores can be generated by compact and scalable on-chip test pattern generators. Furthermore, since written data is unaltered in a fault-free memory, the expected responses can easily be re-generated on-chip and low overhead comparison circuitry can check the correctness of output responses. Therefore, the complexity of memory BIST circuit is lower than that of logic BIST. Due to the deterministic nature and high-test quality of memory test algorithms, memory BIST has emerged as the state-of-the-art practice in industry.

Being parts of an SOC, embedded memories face the same test challenges as SOCs. However, the cost of testing embedded memories has unique characteristics and it is influenced by three major components: cost of ATEs, manufacturing testing time, and DFT and BIST area/performance overhead. When considering the challenges faced by SOC testing, reduced testability, high volume of test data, heterogeneous IP cores and at-speed test, can implementing programmable embedded memory BIST architectures solve all. However, as tens or even hundreds of heterogeneous memory cores are embedded into a single SOC, power-constrained test scheduling is essential to lower the testing time. In addition, a large number of BISTed memory cores (i.e., memory blocks with BIST circuitry around them) will also induce high routing and gate area overhead, as well

as they may adversely influence the memory's speed. Thus, to reduce the overall cost of manufacturing test, it is essential to investigate new memory BIST architectures for complex SOCs, which address the above issues.

This thesis work involves the study; analysis and adding enhance features in such a new-shared BIST architecture developed by the test solution team of STMicroelectronics company. Also major development work has been done for dedicated BIST architecture for an embedded individual memory (single-port SRAM, Dual-port SRAM) .The organization of this thesis and main contributions are summarized in the following section.

1.4 THESIS ORGANIZATION

The major development works carried out in this dissertation work are:

- Development and validation of BIST for single-port SRAM memories with different test options and their combinations.
    - o The development work is done in Verilog Hardware Description Language.
    - o After the BIST insertion in memory the validation of the whole block is done which involves the basic validation steps as RTL simulation, GATE simulation, Synthesis, Formal equivalence check, test coverage analysis. of BIST for single-port SRAM.
    - o According to the requirements of the end user fore such BISTs have been developed and delivered as a part of second phase of this major project work.
- Adding Enhance features in Developed Shared BIST architecture.
    - o Study and analysis of the new Shared BIST architecture developed by the team has been done for area and speed benchmarking of this architecture compared to the existing architecture.

    - Enhanced feature such as Address delay decoder test has been added in the existing RTL code of this architecture. Validation of the whole block is done again.

- BIST compiler development
    - The BIST compiler development work involves UNIX shell scripting and C programming.
    - Study of the whole development procedure has been done as a part of this project work in order to facilitate the team to understand the compiler development work.

The organization of this thesis is as follows. Chapter 2 introduces the basic memory concepts and memory fault models and summarizes the March test algorithm that uses these functional models. Chapter 3 describes the design of BIST for Single-port SRAM memory, which is the key activity of this project work. Chapter 4 summarizes the other contributions done in the project period to facilitate the team for various tasks. Chapter 5 summarizes the results.
Finally, the conclusion and suggestions for further scope of work are given in Chapter 6.

CHAPTER 2

MEMORY TESTING METHODOLOGIES

This chapter introduces the basic theory behind memory testing. There are two kinds of memory test methods: electrical (technology-dependent) and functional (technology-independent). Electrical memory testing consists of parametric testing, which includes testing DC and AC parameters, IDDQ and dynamic testing for recovery, retention and imbalance faults [1]. DC and AC parametric tests are used to verify that the device meets its specifications with regard to its electrical characteristics, such as voltage, current, and setup and hold time requirements of chip’s pins. Since embedded memories in SOCs usually do not have their I/O ports directly connected to chip’s pins, parametric testing for embedded memories is not a necessity. IDDQ and dynamic testing [6] need a detailed description of the specific process technology.

This dissertation work focuses on technology-independent functional memory testing, whose purpose is to verify the logical behavior of a memory core. Because functional memory testing allows for the development of cost-effective short test algorithms (without requiring too much internal knowledge of the memory under test), it is widely accepted by industry as a low-cost/high-quality solution. This chapter provides a theoretical background and explains the memory functional test models and March algorithms. Most of the definitions and figures in this chapter are excerpted from [2, 1].

2.1 MEMORY CONCEPTS

In this chapter we discuss about the basics of memory architecture and it’s functionality. The memory testing will be based upon these fundamentals.

This thesis focuses on technology-independent functional memory testing. A functional model of a memory is based on its specifications. In this model, the internals of the memory are partly visible; hence it is also referred to as the graybox model. One of the main advantages of functional models is that they have enough details of data paths and adjacent wires in the memory to adequately model the coupling faults.

2.1.1 RAM Basics

A RAM is an array of memory cells whose read ports control cell content output and whose write ports control cell content input. A RAM can have any number of read and write ports, with each port having its own separate inputs and outputs. The set of inputs for each read port includes one or more read control lines and N read address lines. A read port’s outputs consist of M data output lines. The number of address lines and data outputs must be the same for all read ports.

The set of inputs for a write port includes one or more write control lines, N writes address lines, and M data input lines. The number of address lines and data inputs must be the same for all write ports. Additionally, the numbers of write address lines must match the number of read address lines, and the number of data inputs must match the number of data outputs.

Address lines identify which column of cells (set of values) to place on the data input or output lines. A RAM can store values into $((2^N)*M)$ memory cells. The read operation places M values at a time on the outputs; likewise, the write operation receives M values at a time on the inputs. Thus, assuming encoded address lines, you can place from 0 to $((2^N)-1)$ addresses on the address lines.

To read a RAM value, you first write a value to the specified location. To perform a write operation, you place the proper address on the write address lines, place the proper data on the data inputs, and activate the write operation (typically, turn on write enable and pulse write clock).To turn on the read operation, activate the read control lines. This places the value stored at the location specified by the address lines on the data outputs. When the read operation is off (not activated), the RAM places X's.

Some memories drive their outputs only when the enable signals are asserted.

2.1.2 Functional RAM Chip Model

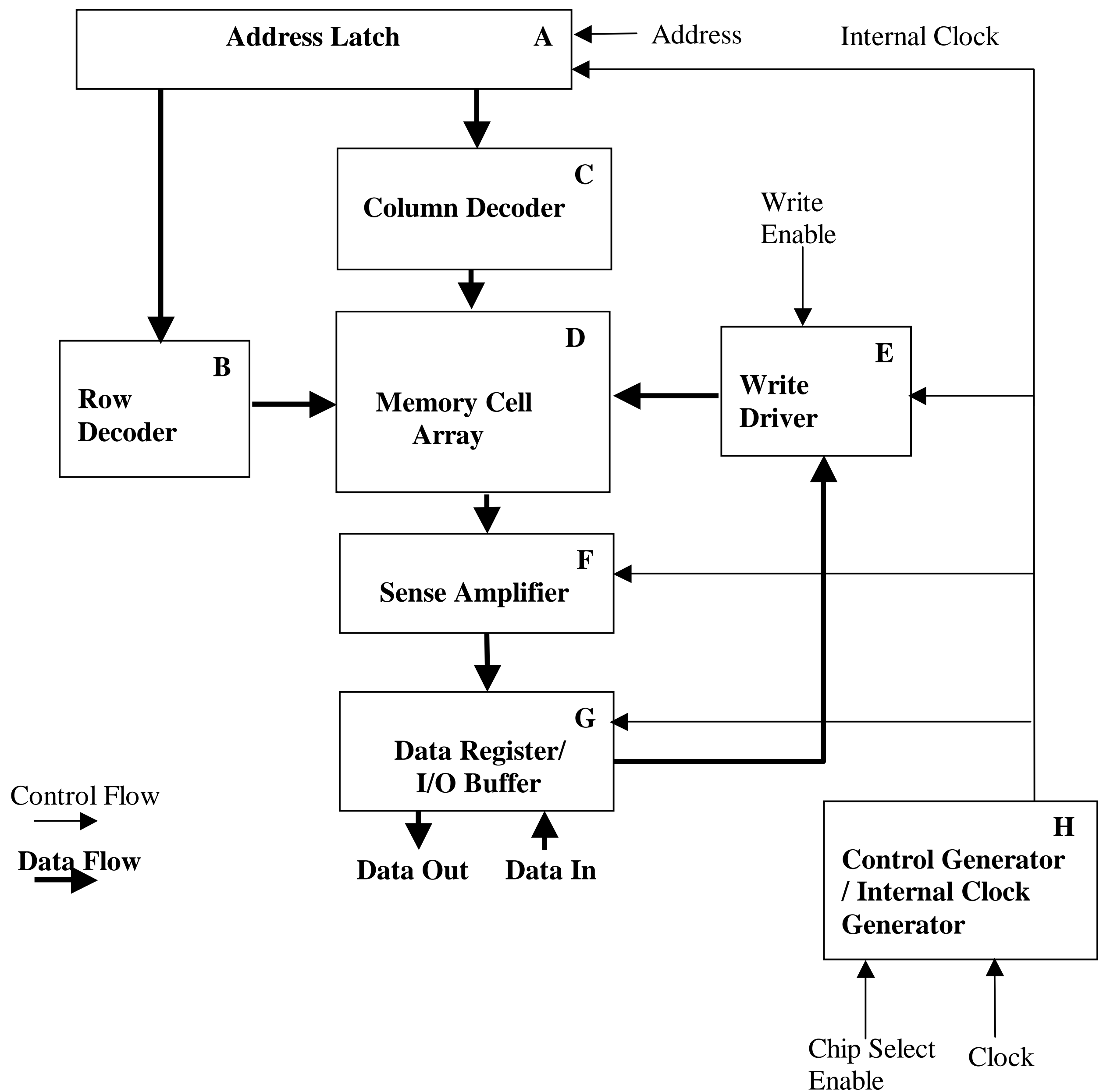


Fig 2.1: Functional RAM chip model [1]

2.1.2.1 BASIC DESCRIPTION OF ABOVE BLOCKS

1. Block'A', address latch, contains the address.

2. The higher order bits of the address are connected to the row decoder 'B', which selects a row in the memory cell array 'D'.

3. The lower order address bits go to the column decoder 'C', which selects the required columns. The number of column selected depends on the data width of the chip, that is the number of data lines of chip, which determines how many bits can be accessed during a read or write operation.

4. When the read/write line indicates read operation, the contents of the selected cells in the memory cell array are amplified by the sense amplifiers 'F', loaded into the data register 'G' & presented on the data-out line(s).

5. During a write operation the data on the data-in line(s) are loaded into the data register & written in to the memory cell array through the write driver 'E'. Usually the data-in & data-out lines are combined to form bidirectional data lines, thus reducing the number of pins on the chip.

6. The chip-enable line enables the data register & together with read/write line, the write driver.

7. Block "H" generates the internal clock with respect to the external clock edge.

2.2 MEMORY MODELLING

Memory modeling is very important in creation of Memory BIST. It provides necessary information to the tool that is used to create the BIST controller. Usually it is provided by

the memory vendor, but if not, can be created also. The memory should be modeled in DFT library format specifically for the memory BIST architecture tool.

2.2.1 ADDRESS AND DATA SCRAMBLING AND DESCRAMBLING

When looking externally at a memory, the words are stored consecutively with respect to the address values, and the data bits within each word are stored in the order of their sequential numbering. This arrangement is called logical mapping of a memory.

In many cases, the physical arrangement of memory cells does not correspond to the assumed logical arrangement, as a result of different memory design requirements. Some reasons for these differences include the following:

- In order to deal with small memory cells, memory designers sometimes fit the periphery cells in the pitch of more than one memory cell. For instance, they lay out sense amplifiers in the pitch of 4, 8, or more memory cells, so they place the corresponding bits of different words next to each other in the memory core, to be able to multiplex these corresponding bits onto one common senseamplifier circuit.
- In order to balance the load on different address lines or (pre)decoded lines, memory designers sometimes scatter the wordlines or bitlines.
- In order to minimize the size of address and column decoders, as well as the length and hence propagation times of row and column select lines, memory arrays are typically divided into several subarrays.
- In order to increase the yield for larger memories, spare (redundant) rows and/or columns are often implemented, which typically disrupt the physical address sequence.

In order to physically preserve the patterns, it becomes necessary to describe the mapping between the physical and logical cell arrangements. The differences between the logical and physical cell arrangements are typically due to a scrambling of the rows and columns and/or data bit lines. This scrambling can be described by a logical transformation or

mapping between the address and data signals required to access the logical memory cell arrangement and those signals required to provide the same data pattern in the physical memory cell arrangement.

2.2.1.1 ADDRESS SCRAMBLING

Frequently in many designs, physically adjacent cells do not correspond to consequent external addresses. That is the memory translates the external address (logical address) supplied to some internal address (topological address) that it uses to access a specific memory cell. This translation is known as address scrambling.
Address scrambling is supported through the address and data scrambling definition portion of the BIST definition inside the memory BIST model.

Address Scrambling Uses:
Address scrambling is important for accomplishing the following-

- General-purpose reduction: - Decoders are often restricted in size in order to fit the following topology of certain cells.
- Increased manufacturing yield: - Extra or left over rows or columns of memory cell can cause a discrepancy between logical and topological addresses.
- Standard Address Pin Assignments: - Address pin number and allocation become standardized, leading to a mismatch between on-chip address pads and standard pin assignments in different designs.

2.2.1.2 ADDRESS DESCRAMBLING

To successfully test interaction between physically adjacent cells in a memory, a dedicated description of the address scrambling function is required in order to generate an address descrambler for testing purpose. The address descrambling block is added in the test path between the BIST controller and the mux.

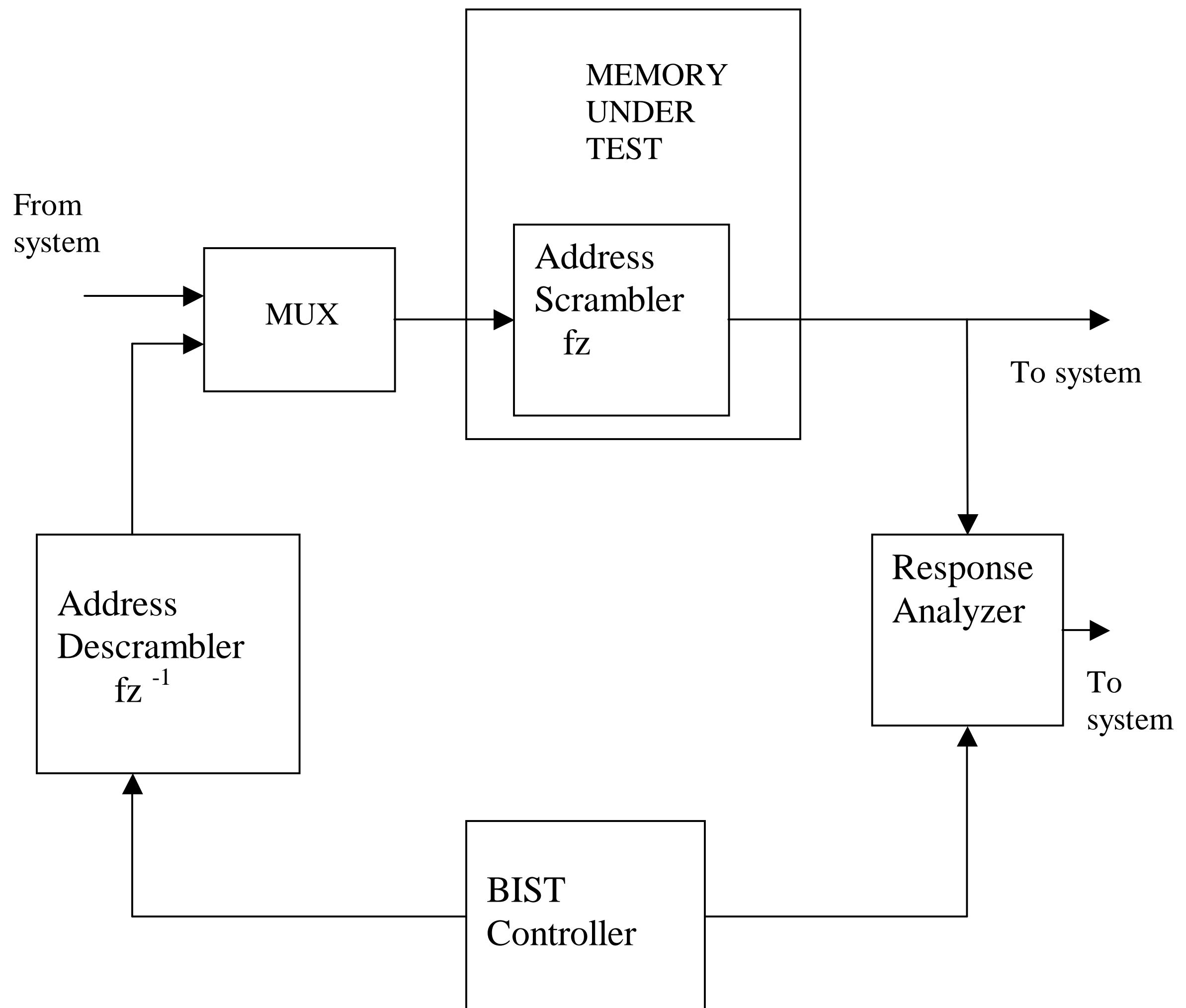


Figure 2.2: Address descrambled block inserted in the test path

2.2.1.3 DATA SCRAMBLING

Memory data is also communicated by a sequence of bits in an external data word (logical data) that might differ from the sequence of bits in data words that physically exists in the memory (topological data). The translation between these bit sequences is known as data scrambling.

Data Scrambling Uses :

Similar to the address scrambling, the data scrambling is important to accomplish the following.

- Space Reduction: - Minimizing the size of column decoder.
- Reduction of Bit/Bit capacitance effects.
- Increased manufacturing yield.
- Standard address pin assignment.

2.2.1.4 DATA DESREMBLING

The data descrambler added by the tool are the same functions as the data scrambling function provided by the memory, this is due to the fact that data scrambling is based upon bit inversion.

Deriving The Address Descrambling Information:

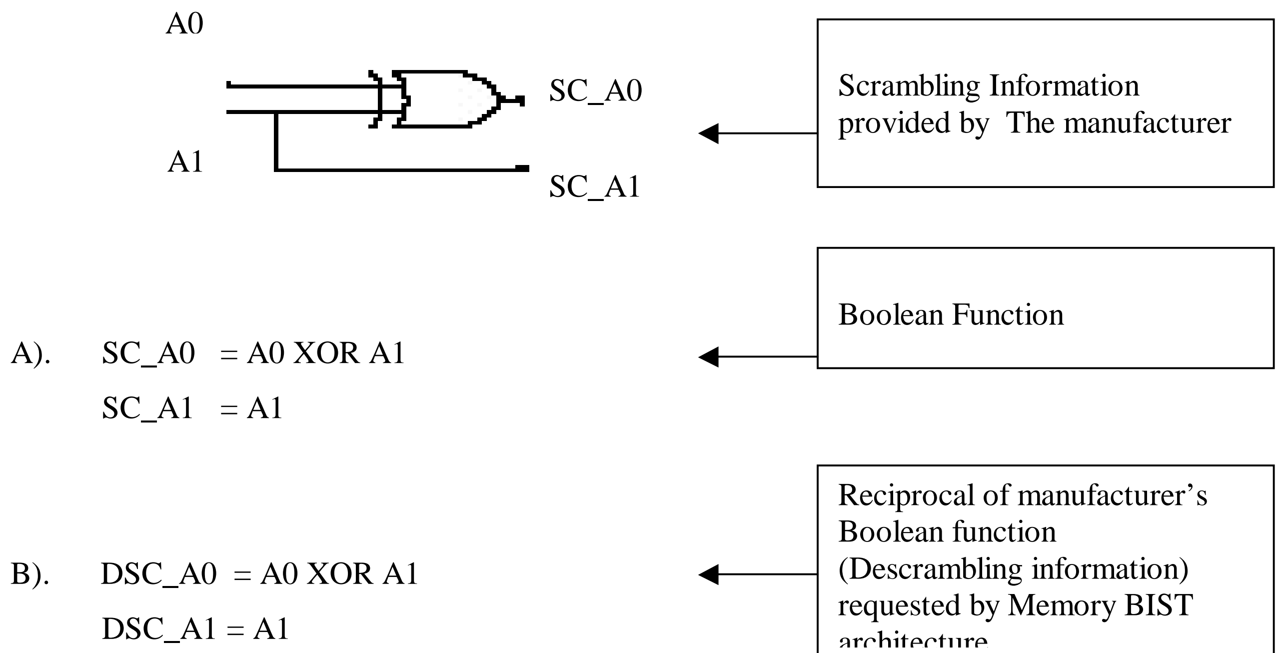


Figure 2.3 Data Descrambling Example

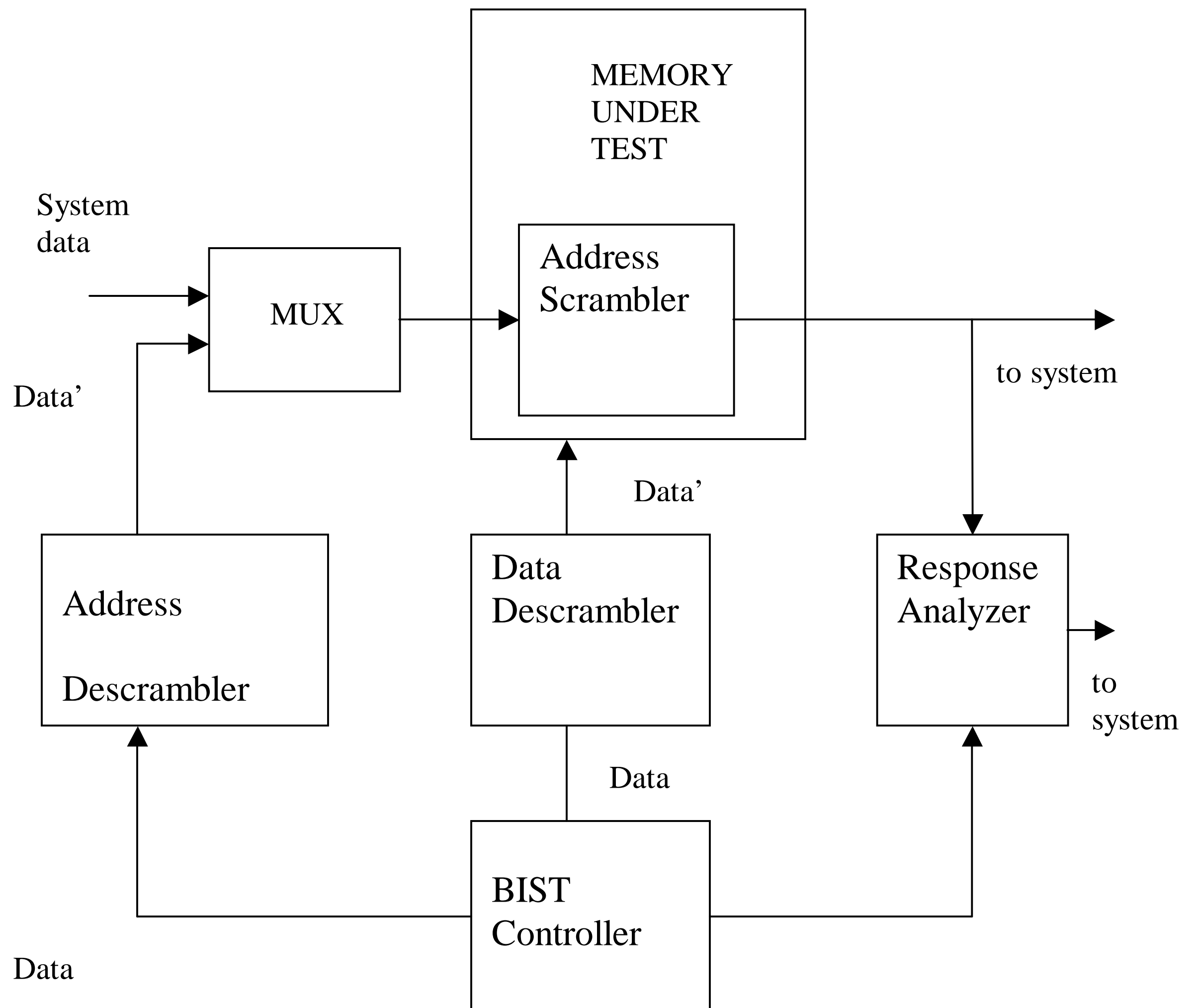


Figure 2.4: Data Descrambler Insertion

It is important to take into account these address and data scrambling effects when testing memories using checker board algorithm which is dependent upon following-

1. Topological address/data information.
2. Consistancy between logical data values and electrical data values.

In order to avoid routing congestion both the descrambler and muxes are implemented close to memory.

2.3 Fault Modeling

Based on the functional memory model shown in Figure 2.1, a subset of functional memory faults are listed in Table 2.1 [1].

	Functional faults
a	Cell stuck
b	Driver stuck
c	Read/write line stuck
d	Chip–select stuck
e	Data line stuck
f	Open in data line
g	Short between data lines
h	Crosstalk between data lines
I	Address line stuck
j	Open in address line
k	Shorts between address lines
l	Open decoder
m	Wrong access
n	Multiple access
o	Cell can be set to 0 but not to 1 (or vice versa)
p	Pattern sensitive interaction between cells

Table 2.1 Functional Memory Faults [1]

In this table, a cell can be either a memory cell or a data register and a line is any wiring connection in the memory. In production manufacturing testing once a fault is detected the memory chip is discarded and no diagnosis needs to be undertaken immediately. Failure analysis through fault diagnosis is performed at a later time and more

comprehensive test sets (using fault-distinguishing patterns) are applied to identify the source of physical defects. Therefore, for production testing, the faults listed in Table 2.1 [1] can be mapped onto the reduced functional faults shown in Table 2.2 [1]. Table 2.3 [1] summarizes the relationship between the functional faults (Table 2.1) and the reduced functional faults (Table 2.2).

Name	**Functional faults**
SAF	Stuck at Fault
TF	Transition Fault
CF	Coupling Fault
NPSF	Neighborhood Pattern Sensitive Faults
AF	Address Decoder Fault

Table 2.2: Reduced Functional Memory Faults [1].

Reduced Functional Faults		**Functional faults**
SAF	a	Cell Stuck
SAF	b	Driver Stuck
SAF	c	Read/Write line Stuck
SAF	d	Chip Select line Stuck
SAF	e	Data line Stuck
SAF	f	Open in data line
CF	g	Shorts between data lines
CF	h	Crosstalk between data lines
AF	i	Address line Stuck
AF	j	Open in address lines
AF	k	Shorts between address lines
AF	l	Open decoder
AF	m	Wrong Access
AF	n	Multiple access
TF	o	Cell can only be set to either 0 or 1
NPSF	p	Pattern sensitive interaction between cells

Table 2.3: Relationship Between Functional and Reduced Functional Faults [1]

For production testing of embedded memories, a great emphasis is placed on March-based test algorithms, since they have high defect coverage with a very reasonable hardware cost.

2.3.1 Stuck-at Faults

The stuck-at fault (SAF) considers that the logic value of a cell or line is always 0 (stuck-at 0 or SA0) or always 1 (stuck-at 1 or SA1). To detect and locate all stuck-at faults, a test must satisfy the following requirement: from each cell, a 0 and a 1 must be read [1].

2.3.2 Transition Faults

The transition fault (TF) is a special case of the SAF because of the fact that once the non-faulty transition occurs, the faulty cell can no longer transition and hence manifests stuck-at behavior. In some cases, however, a coupling fault with another cell (see Idempotent Coupling Faults, Inversion Coupling Faults , and Dynamic Coupling Faults on subsequent pages) can flip the cell's value, thereby masking the stuck-at behavior. For this reason, transition faults must be considered separately from stuck-at faults..

To detect a transition fault, the following sequence of events must occur: A cell or line that fails to undergo a $0 \rightarrow 1$ transition after a write operation is said to contain an up transition fault. Similarly, a down transition fault indicates the failure of making a $1 \rightarrow 0$ transition. According to van de Goor [1], a test to detect and locate all the transition faults should satisfy the following requirement: each cell must undergo an $\uparrow$ transition (cell goes from 0 to 1) and a $\downarrow$ transition (cell goes from 1 to 0) and be read after each transition before undergoing any further transitions.

2.3.3 Coupling Faults

A coupling fault (CF) between two cells causes a transition in one cell to force the content of another cell to change. The 2-coupling fault model [1], which involves only

two cells, is defined as follows: a write operation that generates an ↑ or ↓ transition in one cell changes the content of the second cell. The 2-coupling fault is a special case of the k-coupling fault [1]. A k-coupling fault uses the same two cells as the 2-coupling fault, however it allows the fault to occur only when another $k - 2$ cells are in a certain state.

- The inversion coupling fault (CFin) is a special case of the 2-coupling fault. It means that an ↑ or ↓ transition in one cell inverts the content of the second cell. A test to detect all CFins must satisfy the following condition: for all the cells which are coupled, each cell should be read after a series of possible CFins may have occurred (by writing into the coupling cells), with the condition that the number of transitions in the coupled cells is odd (i.e., the CFins do not mask each other) [1].
- The idempotent coupling fault (CFid) is another particular case of the 2-coupling fault. It means that an ↑ or ↓ transition in one cell forces a second cell to a certain value, 0 or 1. A test to detect all CFids must satisfy the following condition: for all the cells which are coupled, each cell should be read after a series of possible CFids may have occurred (by writing into the coupling cells), in such a way that the sensitized CFids do not mask each other [1].
- The dynamic coupling fault (CFdyn) is a more general case of the CFid. According to its definition a read or write operation on one cell forces the contents of the second cell either to 0 or 1 [2].
- The bridging fault (BF) is caused by a short circuit between two or more cells or lines. It is determined by a logic level rather than a transition write operation. There are two kinds of bridging faults: AND bridging fault (ABF), in which the logic value of the bridge is the AND of the shorted cells or lines, and OR bridging fault (OBF), in which the logic value of the bridge is the OR of the shorted cells/lines.

- In the state coupling fault (SCF) a coupled cell or line is forced to a certain value (0 or 1) only if the coupling cell is in a given state. It is also determined by a logic level.

2.3.4 Neighborhood Pattern Sensitive Faults

A pattern sensitive fault (PSF) causes the content of a cell (or the ability to change the content) to be influenced by the contents of other memory cells, which may be either a pattern of 0s and 1s or transitions in memory contents. The PSF is the most general case of the k-coupling fault, where k equals the number of cells in the memory. There are two types of PSF: unrestricted PSF (UPSF) and restricted (or neighborhood) PSF (NPSF) . For tractability reasons, all the known algorithms are tackling the NPSFs, which can be further divided into three types: active NPSF (ANPSF), passive NPSF (PNPSF), and static NPSF (SNPSF). NPSF testing algorithms are very complex when compared to March test algorithms [1]. However, for certain process technologies, circuit techniques or memory types, such as high-density DRAMs, testing NPSFs may be a requirement..

2.3.5 Address Decoder Faults

Address decoder faults (AFs) represent faults in the combinational logic of the address decoder. Two assumptions are generally accepted: the faults do not introduce sequential behavior in the address decoder and the faults will manifest identically during read and write operations. To simplify the problem, we first consider bit-oriented memories, in which only one bit data is stored in each memory location. The functional faults within the address decoder can be classified into four AFs [1], as shown in Figure 2.3:

:

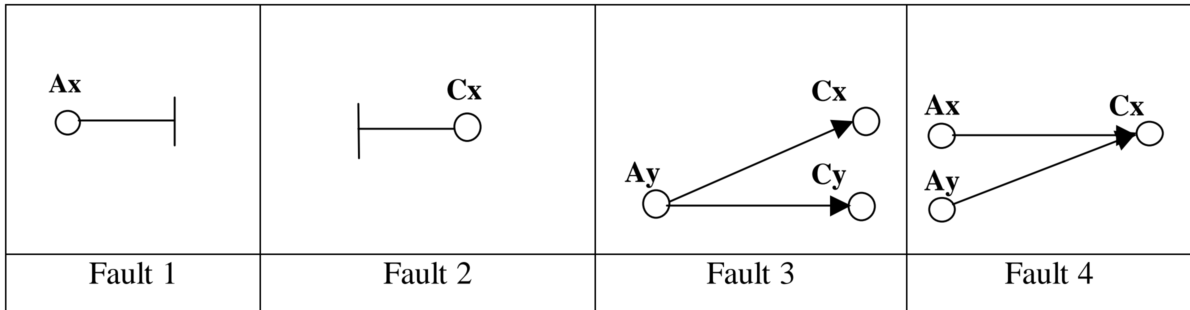


Figure 2.5: Address Decoder Faults [1]

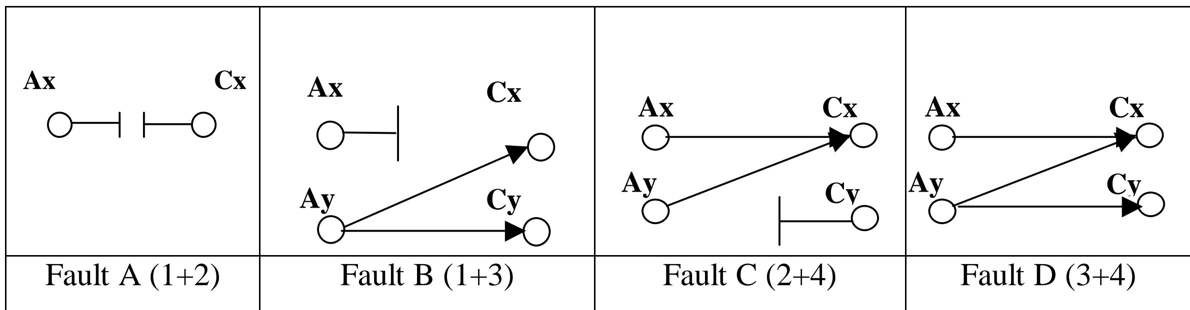


Figure 2.6: Combination of Address Decoder Faults [1]

- Fault 1: For a certain address, no cell will be accessed.
- Fault 2: A certain cell can never be accessed by any address.
- Fault 3: For a certain address, multiple cells are accessed simultaneously.
- Fault 4: A certain cell can be accessed by multiple addresses.

For bit-oriented memories, because each cell is linked to a dedicated address, none of the faults listed above can stand alone. For example, when fault 1 occurs, then either fault 2 or fault 3 will occur as well. Therefore, in total, four fault combinations in the address decoder are shown in Figure 2.4 [1].

2.4 Functional Testing and March Test Algorithms

Based on the used memory fault models, memory test algorithms can be divided into

four categories [1] as described below:

1. Traditional tests including Zero-One, Checkboard, GALPAT and Walking 1/0, Sliding Diagonal, and Butterfly [1]. They are not based on any particular functional fault models and over time have been replaced by improved test algorithms, which result in higher fault coverage and equal or shorter test time.
2. Tests for stuck-at, transition, and coupling faults that are based on the reduced functional fault model and are called March test algorithms [1].
3. Tests for neighborhood pattern sensitive faults.
4. Other memory tests: any tests, which are not based on the functional fault model, are grouped in this category.

As mentioned previously, March test algorithms can efficiently test embedded memories and, therefore, the rest of this section provides more details about them.

March Test Notation

A March test consists of a finite sequence of March elements [1]. A March element is a finite sequence of operations or primitives applied to every memory cell before proceeding to next cell [1]. For example, $\Downarrow$ (r1,w0) is a March element and r0 is a March primitive. The address order in a March element can be increasing ($\Uparrow$), decreasing ($\Downarrow$), or either increasing or decreasing (χ). An operation can be either writing a 0 or 1 into a cell (w0 or w1), or reading a 0 or 1 from a cell (r0 or r1).

2.4.1 Characteristics of March Algorithms

March-based memory test algorithms have several important characteristics:

- Up (down) address sequence must be the exact reverse down (up) sequence, however its internal order is irrelevant. For example, if a 3 bits up address sequence is {0, 5, 2, 3, 7, 1, 4, 6}, then the down sequence must be {6, 4, 1, 7, 3, 2, 5, 0}.

- Most March algorithms are only a simple combination of several March elements. The background pattern during execution can be inferred by the previous operation. For example, a read operation infers the same background data used in the last operation. Similarly, a write operation infers the reversed background data used in the last operation. Based on this observation, one can reduce the number of March elements and the complexity of their implementation. The total number of March elements for the most practical March algorithms is less than ten.
- Using the test generation method proposed in [2], one can generate novel March algorithms (based on a limited number of March elements implemented in hardware), to detect new technology-specific faults.
- For word-oriented memories, one needs to run the March test several times using different background patterns [1, 8] to improve the fault coverage or to use modified March algorithms, such as March-CW [12], to reduce the testing time.

2.4.2 March Algorithms with Diagnosis and Repair Support

When a memory is fabricated using new technology, it is desirable to have a fast yield learning curve [12]. Therefore, it is critical to perform very detailed failure analysis through fault diagnosis to identify the particular defects. The ultimate outcome of failure analysis is a redesigned set of masks for the next fabrication run, which will improve the manufacturing yield. A new set of March algorithms will be used for the best process-specific fault coverage. For large memory chips or SOCs with large embedded SRAMs or DRAMs, to increase the yield, it is crucial to also use redundant memory locations to repair the faulty rows (columns) [12].

A complete solution targeting fault diagnosis and fault location has three components: A memory BIST architecture with diagnosis support to save and send out the diagnostic information, a diagnostic test algorithm and a tool to analyze the collected diagnostic data and generate a detailed fault report for failure analysis and a fault bitmap for repair purposes.

CHAPTER 3

MEMROY BIST METHODOLOGIES

This chapter describes the relevant approaches to embedded memory BIST, summarizes their strengths and limitations and gives the detailed information of the relevant work done in the development of BIST for different memories especially 65nm high speed single-port SRAM.

3.1 Memory BIST Approaches

A typical embedded memory BIST (MBIST) approach comprises an MBIST wrapper, an MBIST controller and the interconnect between them. The MBIST wrapper further includes an address generator to provide complete memory address sequences (i.e., for n address lines all the 2n locations are visited in a complete sequence); a background pattern generator to produce data patterns; a comparator to check the memory output against the expected correct data pattern; and a finite state machine (FSM) to generate proper test control signals based on the commands received from the MBIST controller.

The MBIST controller pre-processes the commands received from upper-level controller (either on-chip microprocessor or off-chip ATE) and then sends them to the MBIST wrapper. The interconnect between the wrapper and the controller could be either serial (i.e., a single command line is shared by all the wrappers) or parallel (i.e., dedicated multiple command lines are linking different wrappers to the controller).

BIST addresses most of the challenges faced by testing embedded memories in an SOC (see Chapter 1 for a full description of SOC testing challenges). However, the increasing size and number of embedded memory cores and the rapid development in VLSI process technologies lead to unique requirements for embedded memory BIST.

1. Support multiple test algorithms: The conventional MBIST approaches usually implement a single March test algorithm. However, deep submicron process technologies and design rules introduce physical defects that are not screened when using the memory test algorithms developed for previous process generations. Therefore MBIST architectures should be programmable to support multiple memory test algorithms to increase the fault coverage and to find the most suitable algorithms for the manufacturing process at hand.

2. Diagnosis and repair support: Diagnosis support in an MBIST architecture is mandatory for manufacturing yield enhancement for new process technology and a rapid transition from the yield ramp phase to the volume production phase [12]. Furthermore, since embedded memories are subject to more aggressive design rules, they are more prone to manufacturing defects (caused by process variations) than other cores in an SOC. For large embedded memory cores, the manufacturing yield can be unacceptable low (e.g., for a 24Mbits memory core, the yield is around 20% [12]). Hence, to achieve a certain manufacturing yield, in addition to diagnosis support, it is also beneficial to introduce self-repair features comprising redundant memory cells.

3. Test heterogeneous memories: State-of-the-art SOCs include many types of memory cores, such as, among others, SRAM, DRAM, flash and ROM. Traditional MBIST approaches were designed to test only one type of memory. However, to reduce area and routing overhead via hardware resource sharing, as well as to decrease the testing time, it is advantageous to develop MBIST architectures that support testing heterogeneous memories simultaneously.

4. Power dissipation constraints: As introduced in Chapter 1, more power dissipation is expected during test mode than power consumed during normal

functional mode for scan-based SOC testing. However, because memory test is functional test, for each memory the power dissipation will be identical in both test mode and normal functional mode. Therefore, if all memory blocks in an SOC can be activated simultaneously during functional mode, power dissipation will not exceed the maximum power constraint during test. Hence, no test scheduling is required in this case. However, to reduce the overall testing time, test scheduling is still necessary for memory testing as described in the following.

On the one hand, for bus-connected memories (BCMs), which are connected to a single-master bus architecture [4], only one BCM can be accessed at any time during functional mode. If all BCMs are wrapped, then all of them can be activated simultaneously during test. Consequently, the power dissipation will be higher during test than during functional operation, and therefore, test scheduling is necessary.

On the other hand, memory testing is part of SOC testing. It was proven in [12] that cores, which use scan-based test methodology, will consume more power during test than during functional mode. If the testing time of these scan-based cores is longer than that of memory cores, then by relaxing the power constraints for scan-based core testing and carefully scheduling memory testing with tightened power constraints, the overall testing time for the SOC can be reduced.

Since test scheduling under power constraints is highly interrelated to the resource sharing mechanisms used in the MBIST architecture, it is essential to develop new power-constrained test scheduling algorithms that will get the maximum usage of the available hardware resources for embedded memory testing.

5. Reuse the available on-chip processing/communication resources: SOCs usually contain one or more processing elements (e.g., microprocessors), which use on-chip system busses to communicate with other cores. Hence the embedded memory cores in an SOC can be divided into two groups: bus-connected memories (BCMs) and non bus-connected memories (NBCMs). Although all the embedded memory cores can be tested by adding dedicated memory BIST wrappers, the high

area overhead of BIST circuitry, as well as the performance penalty caused by intrusive DFT hardware may prove to be the main drawback of this approach. Therefore, reusing the available on-chip resources for testing the embedded memories can lower the area and performance overhead associated with a high number of dedicated MBIST wrappers for BCMs. Furthermore, by implementing non-time-critical tasks in software using a processor, the complexity of the controller can also be reduced.

6. Design reuse: Reusing IP cores in an SOC can greatly simplify the design phase and cut down the time to market. The Reuse Methodology Manual [12] lists various features to make a core reusable. A reusable MBIST core with a scalable and portable architecture, associated with a clear methodology for design flow integration, can significantly reduce the cost of test preparation.

The objective of memory BIST approaches is to meet some or all of the above requirements while reducing the cost of test by targeting low area and performance penalty and low testing time. The existing approaches have explored three main directions to gain improvements: memory BIST architectures, test scheduling algorithms, and special design implementations. Due to their interrelation, without a good architectural support it is hardly possible to achieve any significant improvements through test scheduling or special design techniques. The following sections will review the relevant MBIST approaches presented in the literature.

3.2 Memory BIST Architectures

A memory BIST architecture is defined by the integration of its three components: (controller, wrapper and interconnect). A standalone approach uses a dedicated wrapper and controller for each memory core (or memory cluster with several identical memory

cores), while a distributed approach shares one controller to manage some or all of the MBIST wrappers in an SOC.

3.2.1 Dedicated BIST Methodology

In dedicated memory BIST architecture the BIST is located physically close to the memory so as to minimize routing congestion as well as increasing the speed. The MBIST approach of each memory is independent of other memory's BIST approaches , which makes the implementation of this approach straight forward. However, based on the specific test requirements of different memories and technologies, it needs to be improved in one or more aspects, as described in the following.

MBIST approaches which support multiple March test algorithms are called programmable MBIST architectures . Based on the structure of March algorithms , to support multiple March test algorithms, one can either implement all the March primitives or several March elements. Since there are only four March primitives (r0, w0, r1, w1), by implementing all of them with different combinations of background patterns and address sequences, any March algorithm can be supported. One programmable MBIST approach using March primitives was investigated in [12] and it includes an instruction memory to store the test instructions and a decoding logic to process the test instructions. March element-based approaches implement only several most commonly used March elements. Based on the implemented March elements, only a limited number of March algorithms can be supported.

However, its main advantage lies in less area overhead (simpler decoding logic and less test instructions) when compared to March primitive-based approaches. In addition, by carefully selecting the March elements, new March test algorithms can be generated [2] to target memory faults in new process technologies.

Diagnosis support is another important feature of MBIST architectures. A builtin self-diagnosis (BISD) scheme was introduced in [12]. It sends out faulty memory cell

information (such as faulty address, data, and test session number) for failure analysis. To reduce the control complexity of this approach when testing numerous memory cores, a P1500 MBIST approach with diagnosis enhancement was proposed in [12]. To reduce the testing time in the diagnosis mode (caused by the serial scan-chain structure required to shift out the diagnosis information), a test response compression method was introduced in [10]. Using this method, less I/O pins can be used to send out the faulty response data compared with the uncompressed parallel solution. Due to the increased size of embedded memories, support for memory self-repair is becoming necessary to increase the overall SOC yield. Using the detailed location and information of faulty memory cells (provided by diagnosis support approaches discussed above), one can perform memory redundancy allocation and use fuse-boxes or other methods to repair the faulty memories. However, to collect enough information on fault locations for various memory faults, more complex March test algorithms are required, which implies longer testing time. An MBIST solution was introduced in [12] to test and repair large embedded DRAMs using on-chip redundancy allocation.To reduce the testing time, a memory BIST architecture was proposed in [12] with revised March test algorithms. While most of the previously-described MBIST approaches are focused on testing single port SRAMs, as long as the test algorithms have the features of March algorithms, they are suitable for testing other types of memories with minor modifications. For example, a flash memory BIST architecture was proposed in [12] using a March-like test algorithm. A multiple port SRAM BIST with diagnosis support scheme was introduced in [12] using a modified March algorithm.

In summary, most of the standalone MBIST architectures focus only on solving the test problems related to a single memory core or a standalone memory chip. They do not account for the specific requirements for integrating the design for test hardware for hundreds of embedded memory cores. They also do not provide any support for test scheduling under power dissipation constraints, which needs a flexible control mechanism for the memory BIST hardware.

Figure 3.1 shows the generic Dedicated MBIST architecture :

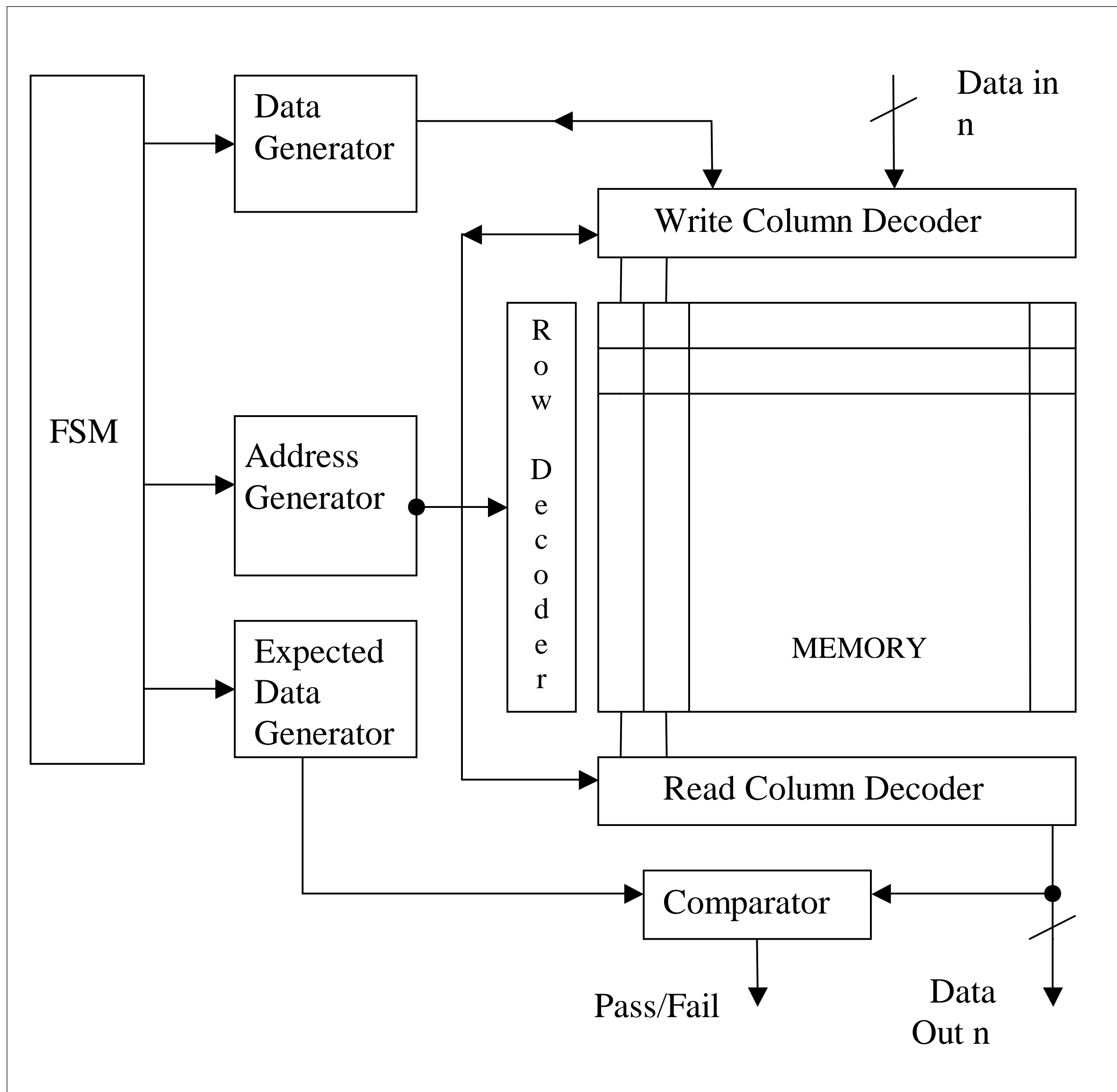


Figure 3.1: Generic BIST Architecture for Standalone memory [1]

The detailed design implementation of all the modules shown in Figure 3.1 can only be described with a specific architecture and it is beyond the scope of this thesis. What are common, however, to most of the known BIST architectures are the comparator, address generator, and background pattern generator in the MBIST wrapper.

1. Comparator: The comparator checks the memory output data against the correct background patterns in order to find any mismatch and its implementation is straightforward. A comparator compares the values read out of the memory with expected values generated by the expected data generation block on a cycle-by-cycle basis. The result of each comparison is accumulated into a status flip-flop in order to provide a go/no-go result at the end of the test. Often, the comparison result is brought out to a chip-pin for real-time monitoring.

2. Address Generator (AG): The address generator for March-based memory testing has several requirements . The most important features of the address generator are that it must cover the entire address space, the internal order of the sequence is irrelevant, however, the down sequence must be in the reverse order of the up sequence. According to these requirements, an automatically synthesized up/down binary counter is sufficient to be the address generator. However, the area of a binary up/down counter is too high for large address spaces [1 , 11]. Linear feedback shift registers (LFSR) [5] may overcome this problem, however, since a traditional LFSR does not cover the all 0s pattern, which is necessary for memory testing, it has to undergo some modifications. Furthermore, the LFSR must also be controlled to generate the reversed (down) sequence. A modified LFSR was described in [1 , 2] to address these two issues. Another address generator was proposed in [11] to reduce the switching activity on the address lines for power reduction. The activity is minimized when two successive addresses differ in exactly one position. This code sequence is known as Gray-code [12]. However, the area of a Gray-code counter (regardless of the implementation type, i.e., FSM-based or conversion from a binary counter [12]) is much larger than that of an LFSRs[2]. In this thesis a ripplecarry adder has been used for address generation which reduces the area.

3. Background Pattern Generator (Data Generator) : Most embedded memories are wordoriented (i.e., they store more than one bit of data in each address location). In [1], the author listed several background patterns for different fault coverages.

Wang and Lee [8] recently presented a hardware implementation for a word-oriented BPG, however, their solution is very complex and has large area overhead. Since there are only $\log_2 N + 1$ states for a BPG, where N is the word-width, we can use a simple FSM to generate all the background patterns very efficiently with much lower area overhead than [8].

4.Finite State Machine (FSM): A finite state machine (FSM) is used to control the overall sequence of events. For example, the FSM determines whether the address counter should be counting up or down or if the data being generated should be a marching 0 or marching 1 pattern.

3.2.1.1 Design Implementation

A BIST block is an offline verification of the memory under test. The basic operation of memory BIST is straightforward: First, the memory is put into a test mode by the use of muxes placed on every data, address, and control line. A finite state machine writes a test pattern to a memory cell, reads it back, and compares it to the original value. If a mismatch occurs, a flag is set to show that the memory cell under test has a failure.

The test hardware for memory BIST includes-

1. A memory BIST controller (FSM)
2. An Address Generator.
    - Address Counter
    - Address Direction : Row Fast / Column Fast
    - Address Increment / Decrement
3. A background pattern inserter or Data Generator for inserting test patterns into memory columns.
4. A MUX circuit feeding the memory during self-tests from the controller.
5. A comparator for response checking.

The address is incremented and the process continues recursively. The process of stepping through the entire memory space can be done multiple times, using different patterns to more fully exercise the memory.

This basic process can quickly detect all stuck-at faults. However, smaller geometry memories are prone to having neighborhood faults caused by particular values of aggressor neighboring cells. Because a cell may have many physical neighbors, test-pattern algorithms that expose these neighborhood faults can become quite complex, with long test times.

A more elaborated figure is shown below-

BIST Methodology

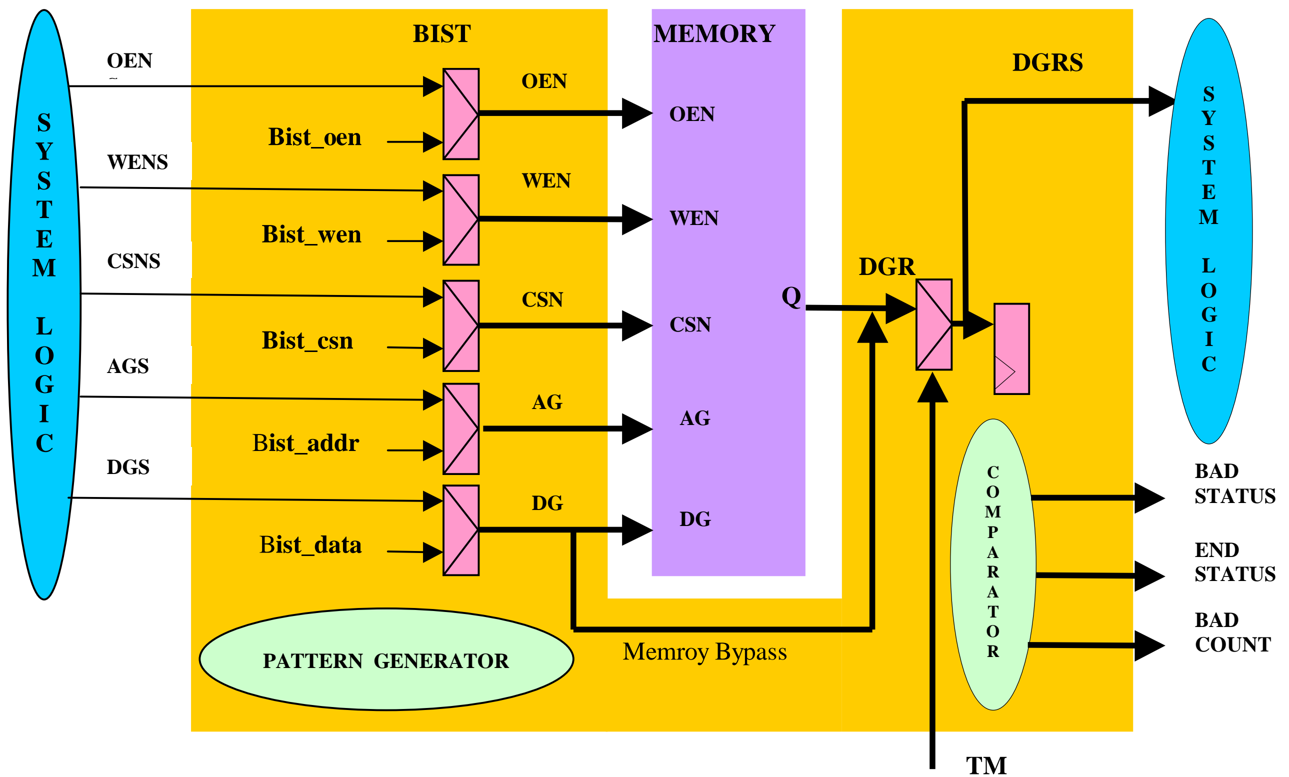


Figure 3.2: Elaborated BIST Execution diagram

3.2.2 Distributed MBIST Architecture

To reduce the BIST area and routing overhead as well as the test control complexity associated with complex and heterogeneous SOCs, distributed approaches are necessary. In a distributed memory BIST architecture, each memory core still has a dedicated technology-dependent wrapper. However, depending on the complexity of the SOC, there are only one (or a few) BIST controllers used to direct the test of all the embedded memory cores. Since hardware resource sharing is introduced, to reduce the routing congestion and to facilitate rapid power-constrained testing, the interconnect between the wrappers and the controller(s) must be carefully considered. Distributed BIST

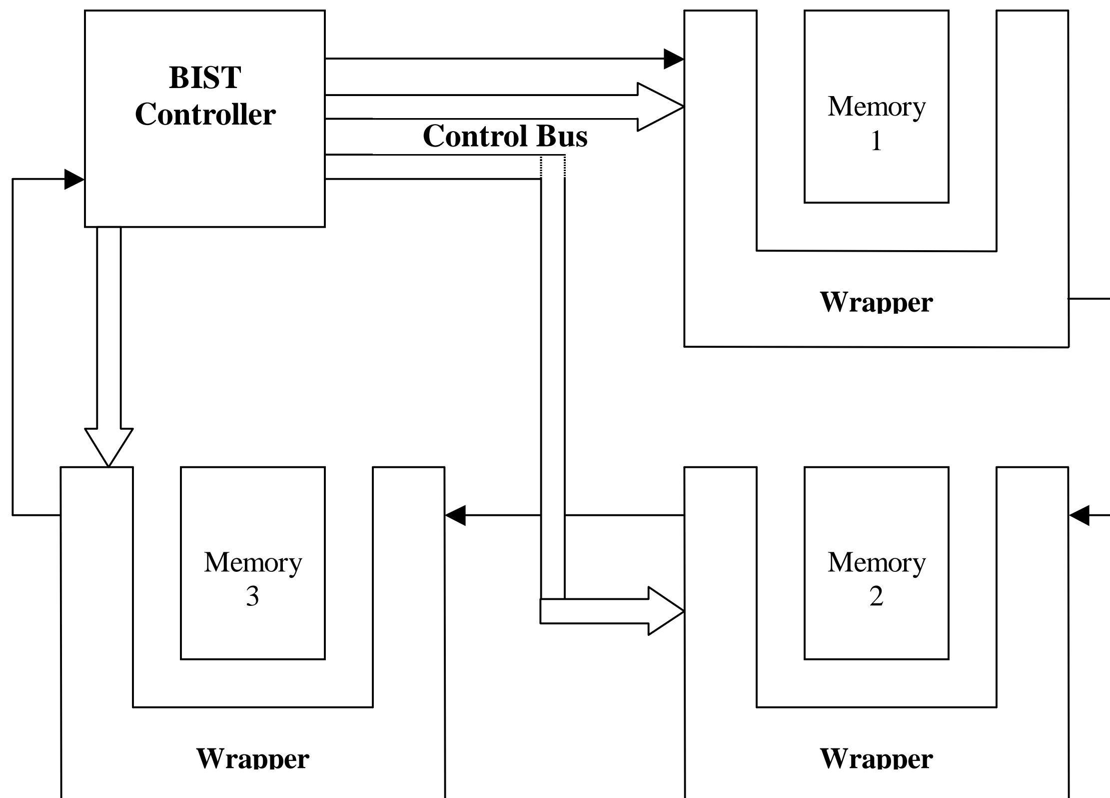


Figure 3.3 : Generic Shared MBIST Architecture

architectures have been advocated for over a decade. Zorian [12] presented a distributed BIST control scheme to test the building blocks of a complex VLSI circuit. Due to the increasing ratio of the memory area in a state-of-the-art SOC, dedicated memory BIST architectures can be used to reduce the cost of memory test.

In Shared BIST methodology ,multiple memories can be tested either sequentially or in parallel. The advantage of testing the memories in parallel is a reduced test time. However, parallel testing has the following disadvantages:

- The power consumption that results from testing several memories together can be high.
- Certain BIST controller resources must be duplicated. For example, a separate comparator is needed for every memory tested in parallel.

There is also a potential disadvantage to using a single controller for multiple embedded memories. If the memories are not localized to one area of the chip, a large amount of wiring might be needed to route the address and data lines from the controller to each of the memories. This disadvantage is especially true for the data lines, given the large data widths (64 bits, 128 bits, and so on) that many memories currently have. To solve this problem, LogicVision's memory BIST offers a patented approach that requires only one data line to be routed to each embedded memory, regardless of the memory's data width. This is referred to as a serial memory BIST approach, because the data values for each test pattern are provided serially to the memory.

CHAPTER 4

BIST FOR 65NM HIGH SPEED SINGLE-PORT MEMORY

This chapter describes the design of BIST for high-speed single-port memory. The execution flow for each test operation has been discussed. On the basis of these flow diagram the RTL code has been written in Verilog HDL for BIST with different test options and their combinations

SPHS65 memory is a high-speed full synchronous single port memory. The main features of the memory are-

- Mux: 4, 8 or 16
- Words: 16 to 16384, in step of 4*mux
- Bits: 4 to 512, in step of one
- Rows: 4 to 1024
- Columns:4 to 1024

4.1 SCRAMBLING

The address bus is split into 2 different parts:

- Row selection: A[n:w]
- Column mux selection: A[w-1:0]

With: $w = \log_2(mux)$

- Here w is always an integer de to the constraint on the mux (4,8 or 16)

4.2 REDUNDANCY

- Two additional rows are present in the bottom of the core of memory to provide reparability.
  - Any two consecutive faulty rows can be replaced with redundant rows.
  - If the faults are not at consecutive rows, the memory will not be repairable.

Note: This constraint is used as in most cases the faults occur due to misconnections developed during fabrication of Via's. Wrong connection in one via causes fault in two consecutive cells, which share the same via for metal connection. Thus if there is a fault in one particular row then there will be possibility of faults in its neighboring rows/columns depending on the memory layout.

4.3 FAULT MODELS of single-port high speed SRAM

- Stuck At Faults
- Stuck Open Faults
- Address Decoder Faults
- Sense amplifier faults
- RY functional faults
- Coupling Faults between Global bitlines with local bitlines.
- Transition coupling faults
- Linked Coupling faults
- Disturb Fault

4.4 ALGORITHM

During the BIST run an algorithm is executed which is composed of a sequence of events in order to cover the different modelized faults inside the memory. For single-port memories the most widely used algorithm is 14nMarchLR algorithm.

4.4.1 mMARCHLR 14N

The mMarchLR 14N algorithm is based on the basic 14nMarchLR 14N described below-

Address Number	S0 Addresses Increased	S1 Addresses Decreased	S2 Addresses Increased	S3 Addresses Increased	S4 Addresses Increased	S5 Addresses Increased
O	WB		RIWBRBWI	RIWB	RBWIRIWB	RB
1	WB	RBWI	RIWBRBWI	RIWB	RBWIRIWB	RB
…	…		…	…	…	…
…	…	RBWI	…	…	…	…
K	WB	….	RIWBRBWI	RIWB	RBWIRIWB	RB
…	…	….	…	…	…	…
N-1	WB	RBWI	RIWBRBWI	RIWB	RBWIRIWB	RB
N		…	RIWBRBWI		RBWIRIWB	RB
	WB	RBWI		RIWB		
		RBWI				

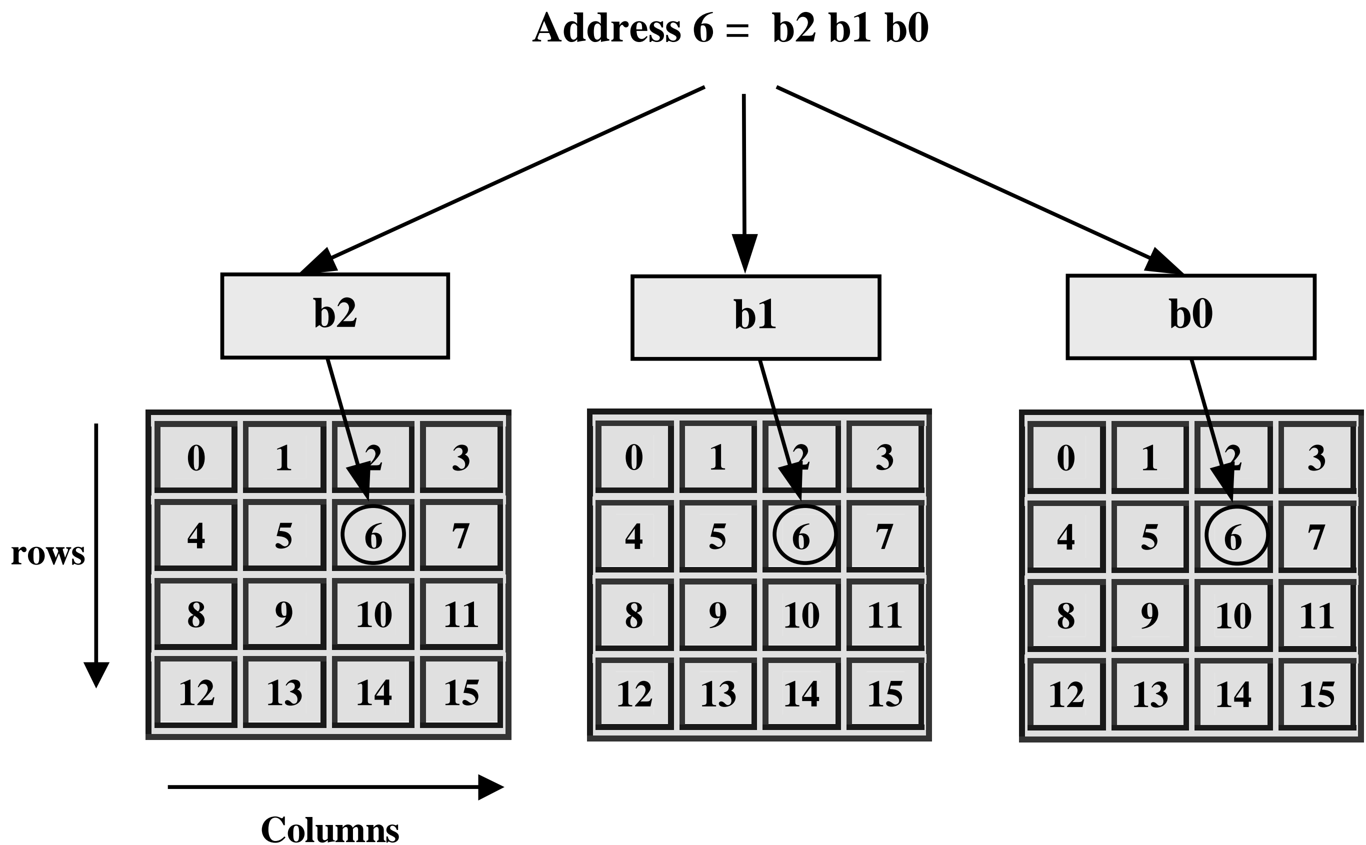
Table 4.1: Modified MarchLR algorithm

Here the different symbols have following meaning-

- wB: Write background
- wI: Write inverted background
- rB: Read background and compare to the expected value
- rI: Read inverted background and compare to the expected value.

The modified MarchLR 14N is an additional layer to the MarchLR 14N. It takes in consideration the physical organization of the memory, and works not with a logical addressing, but with a physical addressing. One more thing is the written data, which change during the algorithm execution inside the different stages to create a physical map. This is what is termed as the address scrambling and data scrambling.

The mMarchLR 14N increases the fault coverage because it creates more layout configurations, susceptible to highlight memory defects. This algorithm is adapted to interleaved memory type (also called bit oriented memory), using a multiplexer to dispatch the words.

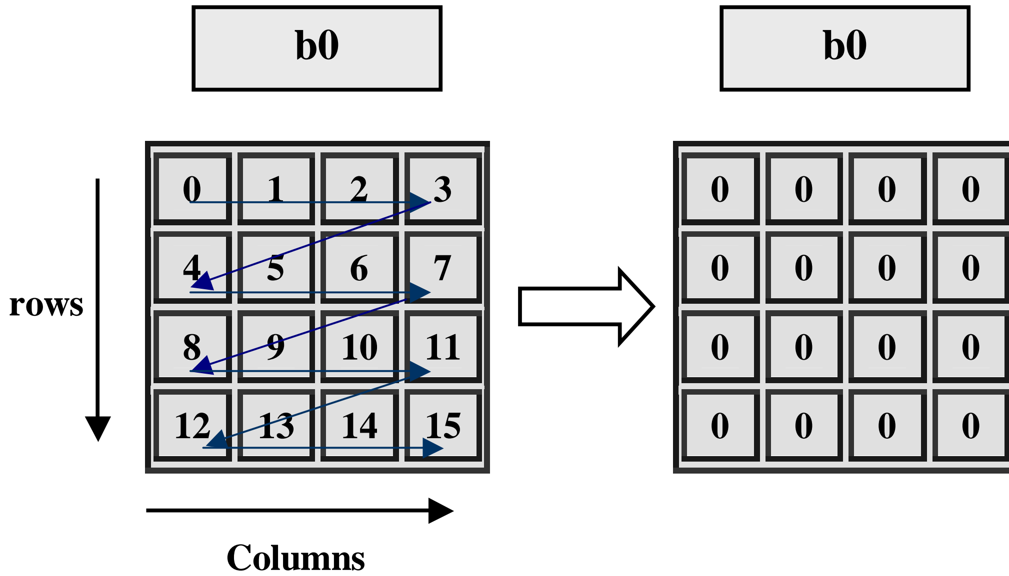


Example of a 16 words of 3 bits memory page. The mux factor is 4 inside a page.

The algorithm is run four times for each physical layout configuration defined below.

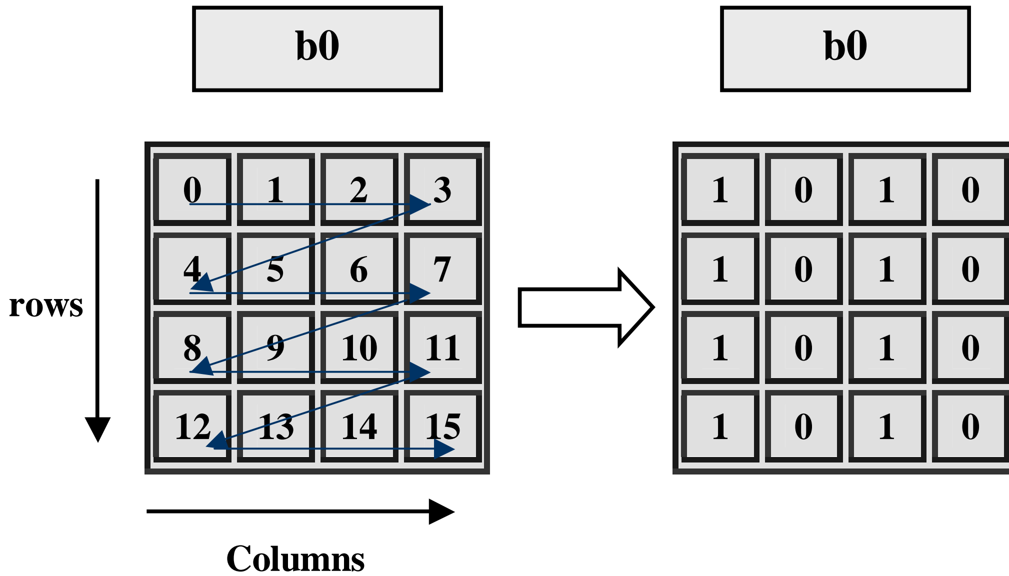
1. Solid

All the memory is filled with 0 (wB). The column address is incremented first.



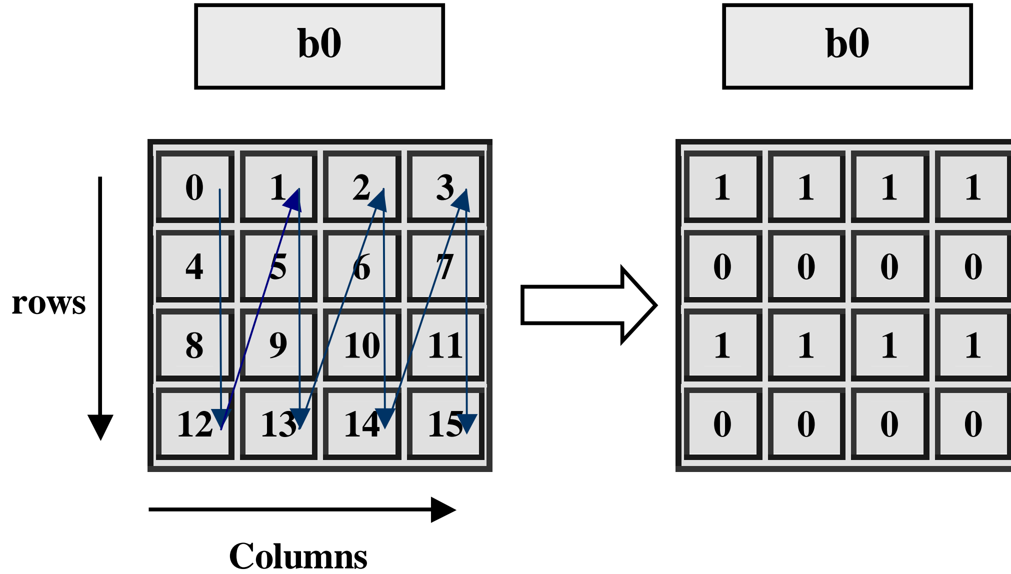
2. Column stripe

The address scrambling of solid is kept, but the data scrambling will create an alternate column stress.



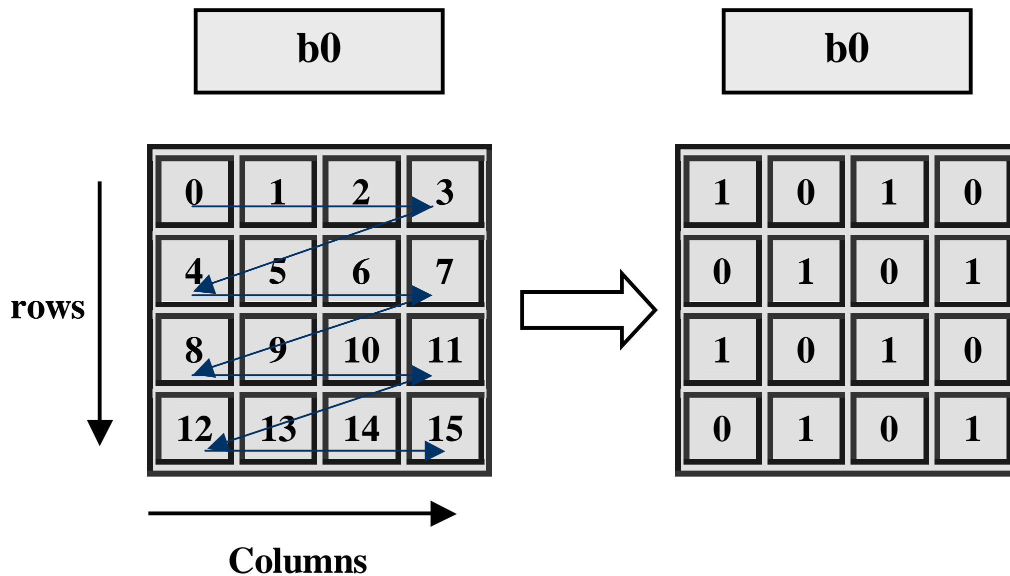
3. Row stripe

The address scrambling increments the row fast in this configuration. The goal is to create an alternate pattern in the row direction.



4. Checkerboard

With the columns stripe address scrambling, the final layout configuration is a checkerboard.



The total number of cycle necessary to execute fully the algorithm:

N cycles = 14.N.4

Where N is the number of words in the tested memory.

The faults that are detected by this algorithm are –

- Address Faults
- Stuck at Faults
- Coupling faults
- Linked Coupling faults
- Some Neighborhood Pattern Sensitive Faults.

4.4.2 MASK BITS TEST

The Mask Bits Algorithm consists of 12 stages-

Address	S1	S2	S3	S4	S5	S6
any	WRITE D0101 M0000 Qxxxx	READ D0101 M1010 Q0101	WRITE D1010 M0101 Q0101	READ D1111 M0000 Q1111	WRITE D1010 M0000 Q1111	READ D1010 M0101 Q1010

Table 4.2: Mask Bits Algorithm [3]

Address	S7	S8	S9	S10	S11	S12
any	WRITE D0101 M1010 Q1010	READ D1111 M0000 Q1111	WRITE D0000 M0000 Q1111	READ D1111 M0000 Q0000	WRITE D1111 M1111 Q0000	READ D1111 M0000 Q0000

Table 4.3: Mask Bits Algorithm (continued..) [3]

The faults detected by this test are-

- Stuck at zero.
- Coupling fault between M bit neighborhood pins
- Coupling faults detected between D bit neighborhood pins
- Coupling faults detected between Q bit neighborhood pins
- Coupling between D,M,Q bus of memory

4.4.3 DATA BIT COUPLING TEST

This test is done to detect the coupling between the data pins of the memory. This test consists of 4 stages-

Address	S1	S2	S3	S4
Any	WRITE D0101 QXXXX	READ D1010 Q0101	WRITE D1010 Q0101	READ D0101 Q1010

Table 4.4: Data bits coupling algorithm

This test takes 4 cycles to complete, and is usually run at one address at the end of the main algorithm.

The faults detected by this algorithm are-

- Stuck at fault
- Faults detected by the Data line coupling faults

4.4.4 CSN BIT COUPLING TEST

This test is done to check the faults on CSN pin of the memory.

S0	S1	S2	S3	S4
WRITE WEN0 CSN0 D0000 QXXXX A…..00	FALSE WRITE WEN0 CSN1 D1111 QXXXX A…..00	READ WEN1 CSN0 D1111 Q0000 A….00	WRITE WEN0 CSN0 D1111 Q0000 A….00	FALSE READ WEN1 CSN1 D1111 Q0000 A….00

Table 4.5: CSN bit Coupling test

This test is completed in 5 clock cycles, and is usually run one time at one address at the end of the 14N MarchLR checkerboard algorithm run.

The faults detected are:

- Stuck At zero on CSN bit of the memory

4.4.5 Address Delay Decoder Test

This specific test is used to highlight delay fault in address decoder by stressing it. Using algorithms specially designed for this test, it can be tested whether address decoders are fast enough while jumping from one address to another.

In other words, the purpose is to see if the address decoder can make the jump from the base address (A1) to the target address (A2), which is offered one cycle later where A1 and A2 are generated as described below.

In this algorithm, the base address (A1) will be generated from zero to maximum address (MAXADDR) in incrementing order, and the target address (A2) will be generated by taking complement of the base address (A1) as and when needed, depending on the operation.

4.4.5.1 Fully Decoded Memory

In case of fully decoded memory, the base address will be generated from zero to maximum address (MAXADDR) in incrementing order and the target address (A2) will be generated by taking complement of the base address (A1).

Example : In case of a MAXADDR of 1111 , the jumping table will be-

A1: 0000 A2: 1111

A1: 0001 A2: 1110

....

A1: 1111 A2: 0000

The sequence of operations applied will be-

- **Fill 1 (All addresses) Initially**
- **WB at A1**
- **RIB at A2**
- **RB at A1**
- **WIB at A1**

4.4.5.2 Not Fully Decoded Memory

In case of not fully decoded memory when we take the complement of the base address A1, the last addresses are skipped (which are greater that the MAXADDR specified) and the complement of msb of the base address is taken to be zero.

Example: In case of a MAXADDR of 1100 the addresses from 1101 to 1100 will be skipped and the jumping table will be-

A1: 0000 A2: 0111
A1: 0001 A2: 0110
…….
A1: 0011 A2: 1100
A1: 0100 A2: 1011

And the sequence of operations will be-

- **Fill 1 (All addresses) Initially**
- **WB at A1(0000), A2(1100)**
- **RIB at A2(0111)**
- **RB at A1 (0000)**
- **WIB at A1(0000)**

4.5 IMPLEMENTATION

4.5.1 I/O Pin Description:

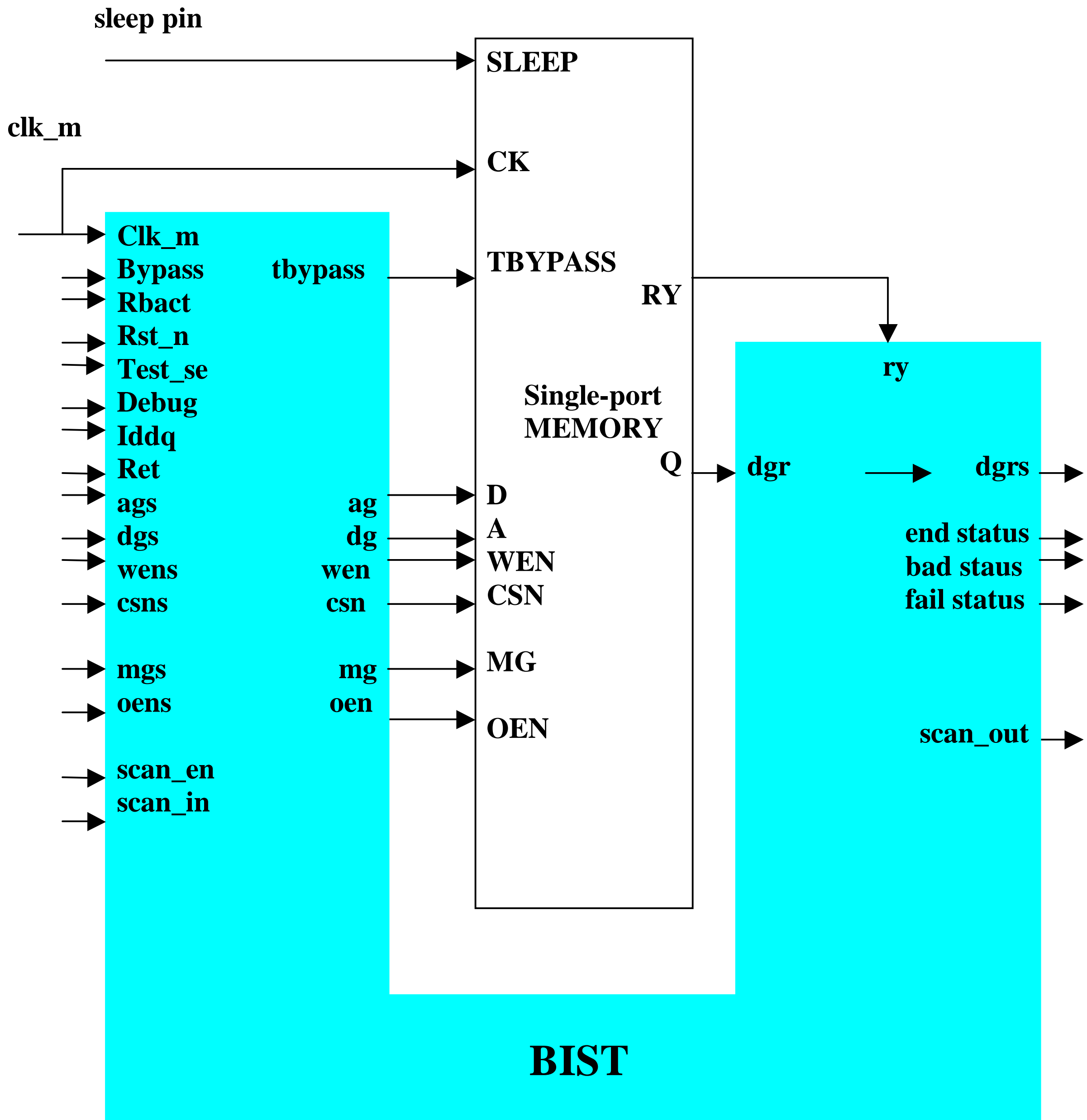


Figure 4.1: Pin connections (BIST AND MEMORY) [3]

4.5.2 The Full Pin Description:

NAME	DIRECTION	FUNCTION
ag	OUT	Address bus to memory
ags	IN	Address bus from design
dg	OUT	Data bus to memory
dgs	IN	Data input from system logic
dgr	IN	Data output bus from memory
dgrs	OUT	Data output to system logic
csn	OUT	Chip select signal to memory
csns	IN	Chip select from system logic
wen	OUT	Write enable signal to memory
wens	IN	Write enable from system logic
tbypass	OUT	Memory bypass control from memory
debug	IN	Debug mode pin
ret	IN	Retention test pin purpose
iddq	IN	Iddq mode pin
bad status	OUT	BIST test failed
fail status	OUT	BIST word failed again
end status	OUT	BIST test end signal
clk_m	IN	BIST clock
bypass	IN	Test mode pin
rbact	IN	Run BIST active to launch the BIST
test_se	IN	Scan chain enable
rst_n	IN	BIST reset
Tristate output		
oens	IN	Output enable from system logic
Redundancy		
rras	OUT	The row repair address from system
rraes	OUT	The row repair address enable from system
rra	OUT	The row repair address to ram
rrae	OUT	The row repair enable to ram
Scan_en	IN	Shift out enable
Scan_in	IN	Shift in
Scan_out	OUT	Shift out
Bitmap		
clk_bmp	IN	The bitmap clock
Rst_bmp_n	IN	The bitmap reset
bmpout	OUT	The bitmap shift out

4.5.3 OPERATING MODE DESCRIPTION

4.5.3.1 Scan ATPG Mode:

During the scan mode, all the design is tested by a set of vectors generated by an automatic Test Pattern Generator (ATPG). At this stage of the flow, the state of the memory is unknown (bad or good memory which can be a black box). Then it's necessary to switch the BIST in the scan mode which bypasses the memory and activates observability points to increase the fault coverage.

The goal of memory bypass is to test independently design logic and memory. This mode allows user to run the atpg pattern on his chip, bypassing the memory output and hence avoiding the ram shadow i.e. without being impacted by the memory yield. It is important for debug capability.

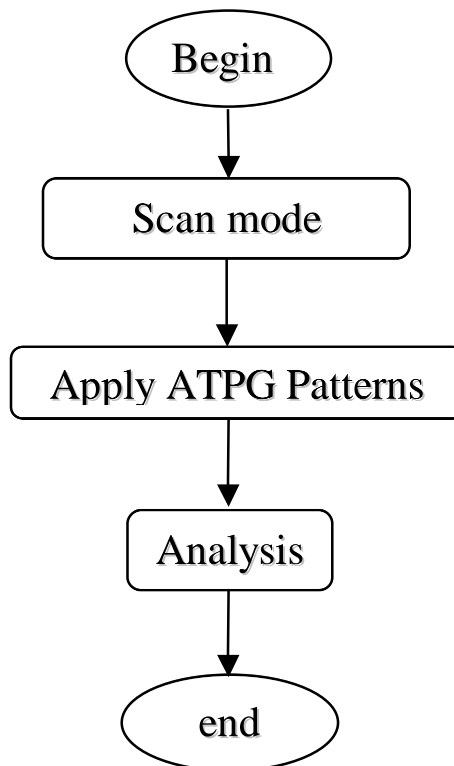


Figure 4.2: Scan ATPG Flow

4.5.3.2 Transparent Mode:

This mode allows user to access the memory directly in the functional mode. In this mode the memory pins are connected to the system pins and the BIST pins are deselected. Inherently the rbact pin decides between the functional mode and BIST active modes. This saves a lot of glitches on the memory pins.

4.5.3.3 Run BIST for RAMs

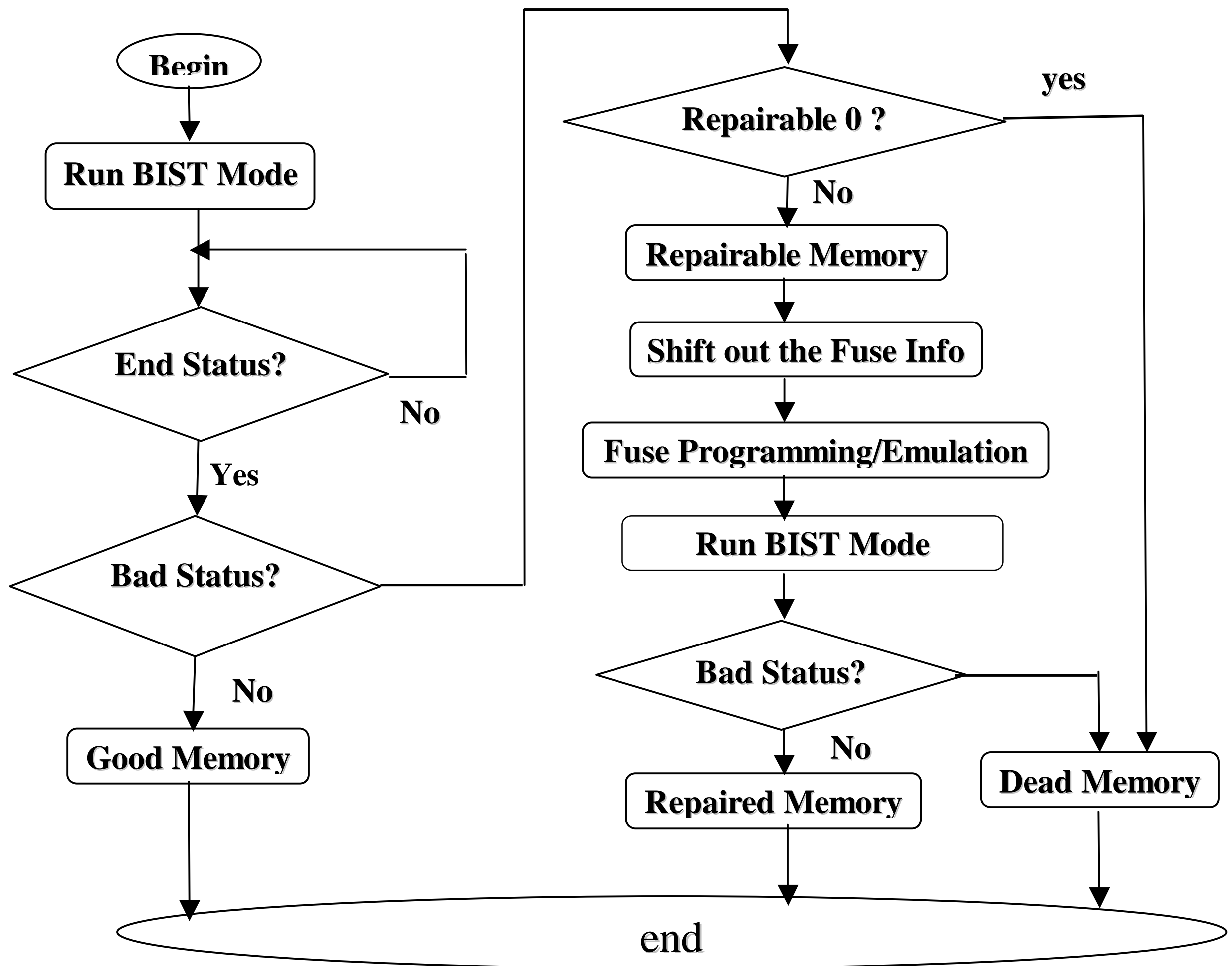


Figure 4.3: Run BIST Execution

The Run BIST program is generated through a synchronous FSM which runs at the same clock as memory under test. The high speed is achieved through pipelining in FSM stages.

During Run BIST mode a sequence of operations are done on memory according to the chosen algorithms.

The different test performed during run BIST for single-port SRAM are:

- Main algorithm run
- Data bit coupling test
- CSN bit coupling test
- Address decoder test

4.5.3.4 IDDQ Fill 0/Fill 1 Modes and Retention Test:

Fill 0/1: The IDDq fill 0 and IDDq fill 1 are useful and fast to write respectively 0 and 1 in the entire memory array especially for the IDDq test.

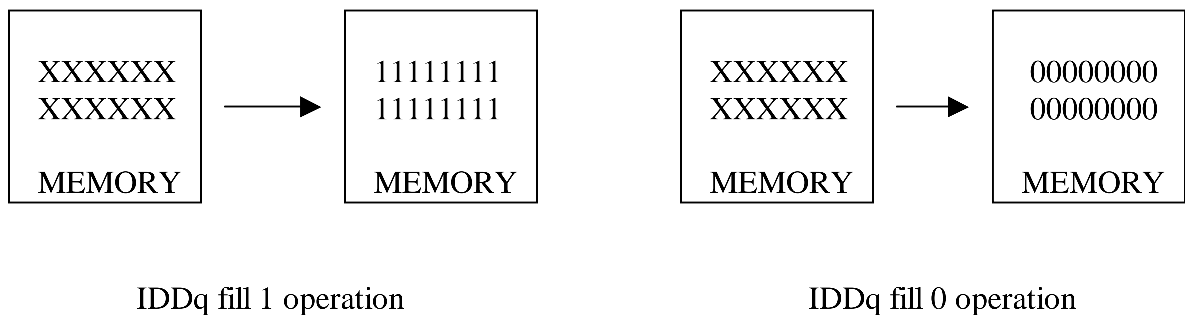


Figure 4.4: Fill 0 and Fill 1 operation

Read 0/1 : The Read 0 and Read 1 features are intended to test the memory retention after a delay defined on tester.

The goal is to check the entire memory to check if the memory contents is set at 0 or 1.

Retention Test: This test is intended to detect the memory cells, which fail to retain a written value after a period of time. This type of faults cannot be detected by the main algorithm of the memory BIST because of the large amount of time requested to highlight this kind of problems.

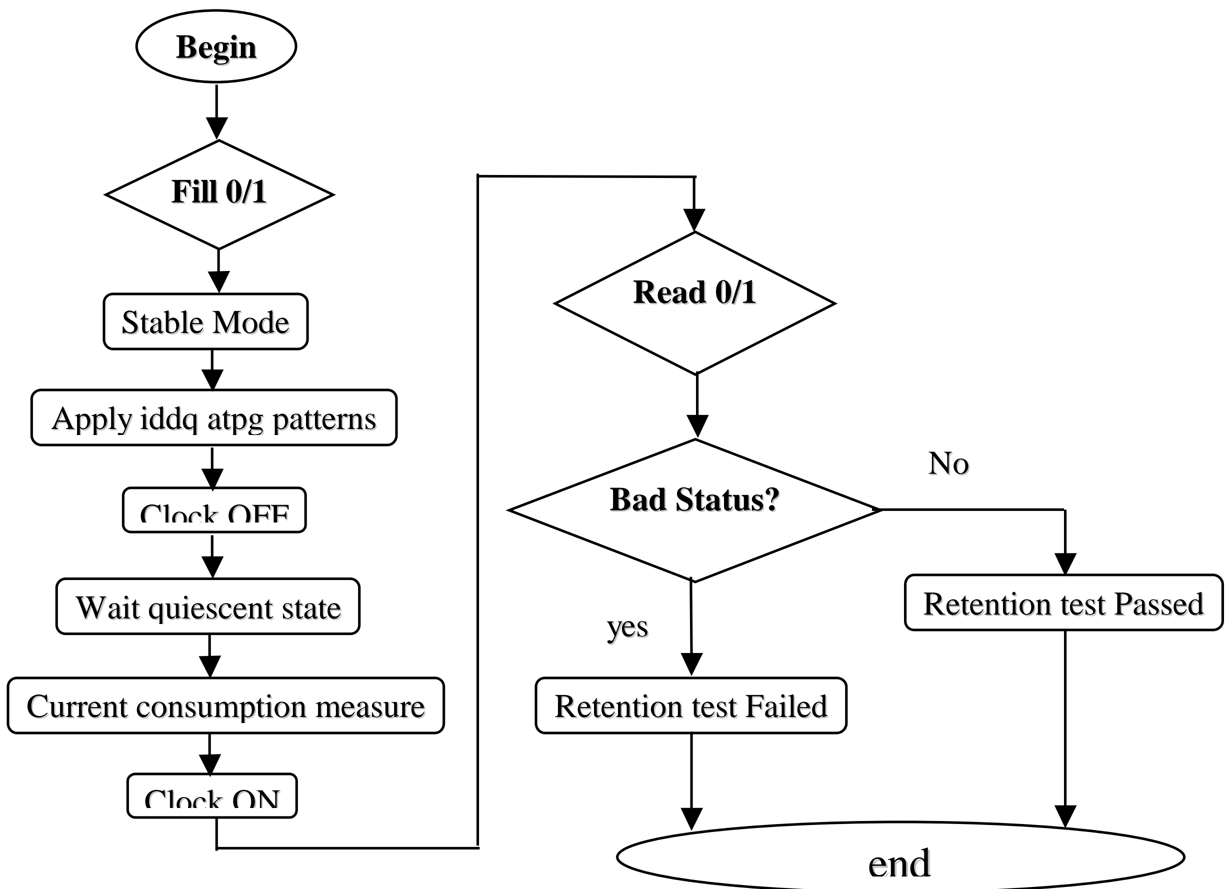


Figure 4.5: Fill 0/1 and Retention test flow

4.5.3.5 Scan Collar Mode

The memory inputs will be controlled by the BIST. BIST will keep the memory inputs like csn gated with test_se (scan chain enable pin) Hence when test_se is 1 no operation will happen on the memory. During capture modes when test_se goes low, any operation will be possible on the memory.

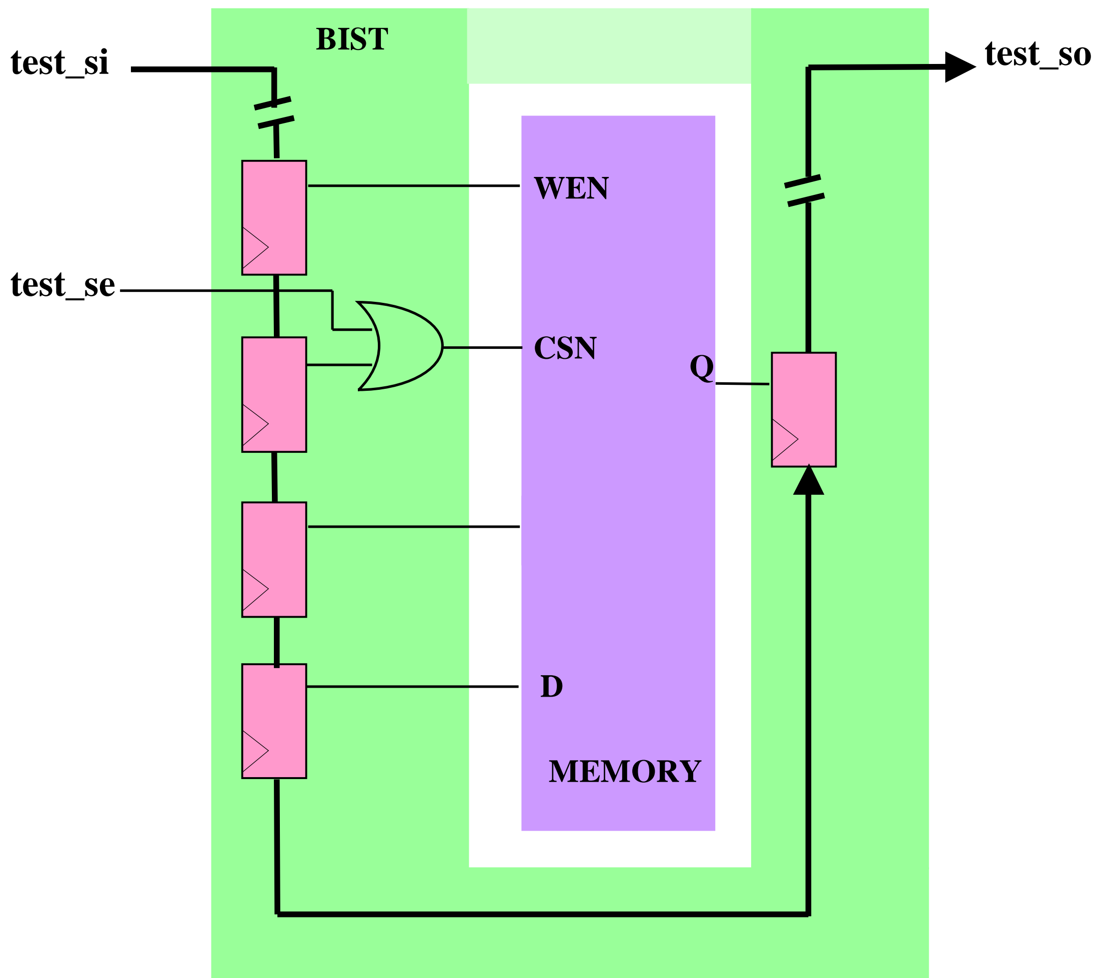


Figure 4.6: Scan Collar Implementation

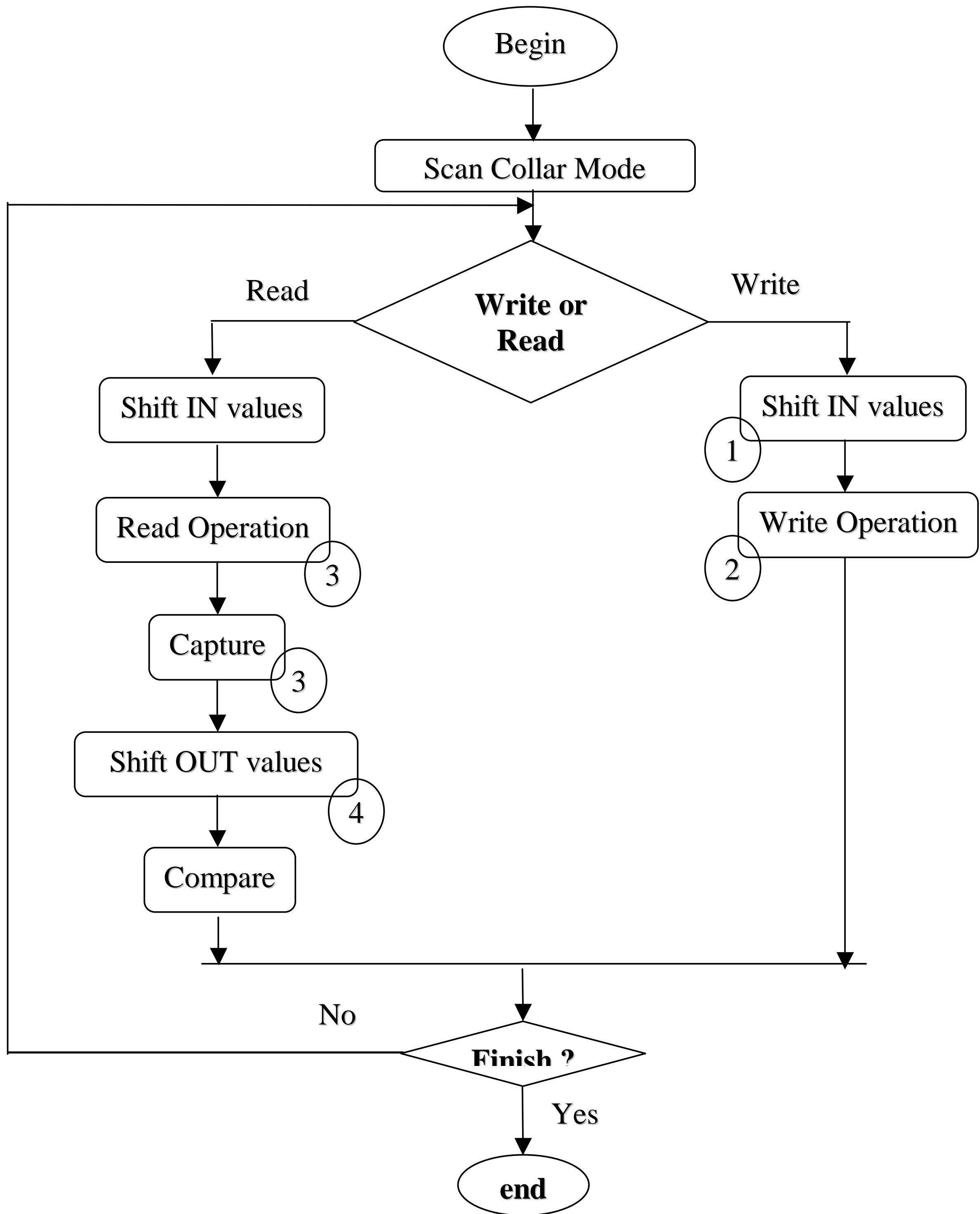


Figure 4.7: Scan Collar Mode test Flow

1. Shift In

At this stage the CSN and WEN values are shifted in scan chain along with other values. The test_se pin is kept high during this mode to disable any operation on memory.

2. Write Operation

During write operation WEN= 0 and CSN = 0. The write operation can't be disassociated with the shift in operation, because it's one of the two possible continuations. test_se goes low to allow write operation on memory.

3.Read Operation and Capture

The read operation is similar to the write operation. CSN = 0, WEN = 1

Once the data are outputed from the memory, it's necessary to save them into the scan registers to shift them out. This very important operation is called Capture. During this stage, test_se pin remains low during one clock cycle.

4. Shift out values

This stage is done just after a read and capture operation. The memory data output is serially shifted out, msb first.

4.5.3.6 BITMAP MODE

The BITMAP feature is a powerful feature to easily diagnose a memory by creating a memory error map. It's an additional layer to the classical RUNBIST.

During a BITMAP mode, the BIST is run.

- Each detected fault, the faulted addresses, datas and BIST parameters are stored into the FIFO, and the tester read, through the shift out register, the information about this fault.

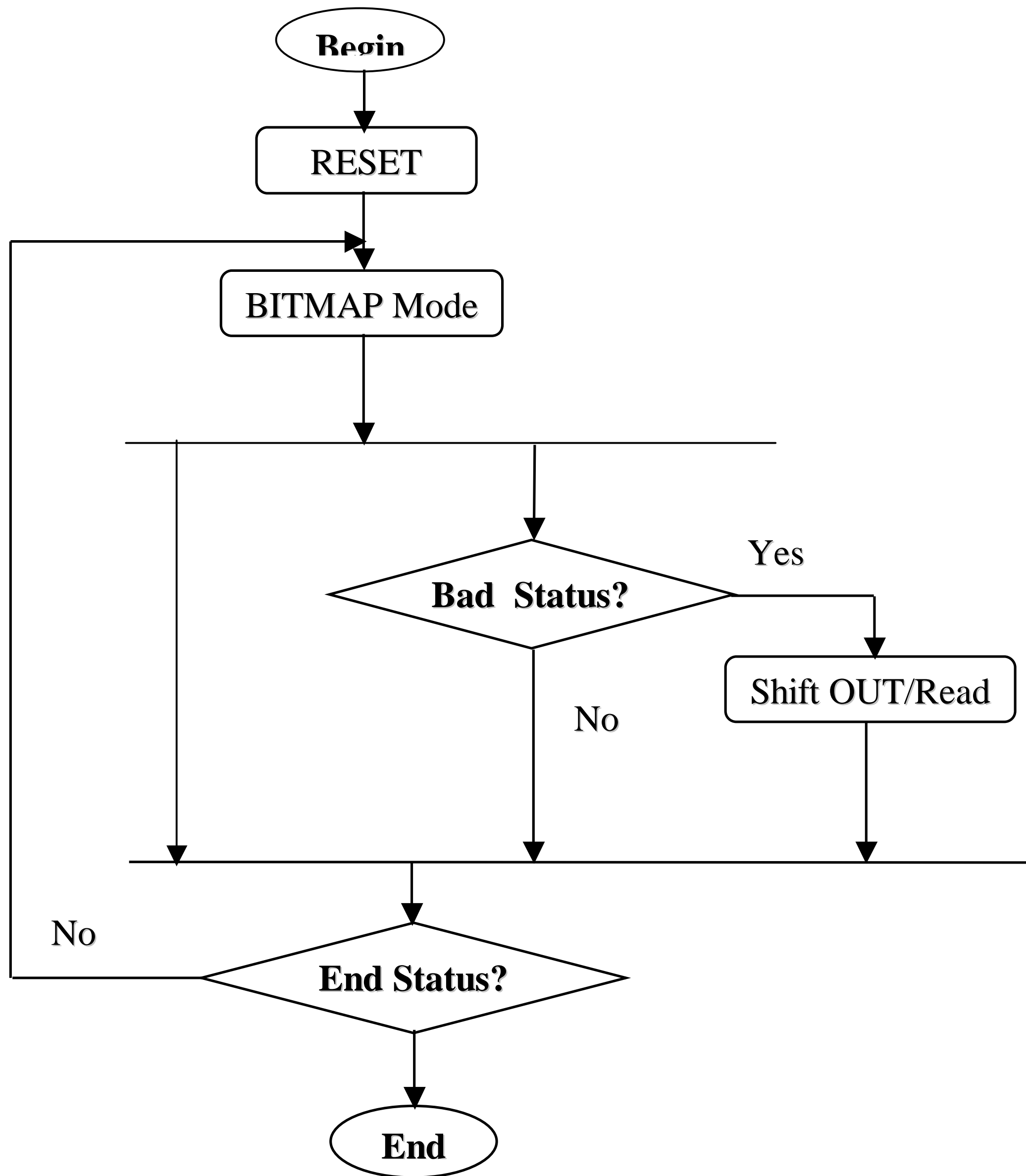


Figure 4.8: BITMAP Mode Test Flow

- The information is shifted out by the tester at a lower speed, without disturbing the correct execution of the BIST.

- This fifo stack works like a buffer between the BIST and the tester to avoid problem of clock speed.

Each FIFO stack contains:

- Address: the address where a fault occurs.
- State: Machine state value where the memory fault is detected.
- Pattern: describe the type of the pattern used to create the fault. This register had a fixed size of 2 bits.
- Data: Content of the memory faulty data.

CHAPTER 5

VALIDATIONWIZARD & RESULTS

After the BIST insertion in the functional model of the memory .The whole design is to be validated to ensure the correctness of BIST designed measured according to following criteria's-

- Functional correctness of BIST at different levels of RTL.
- Area overhead measurement on the chip because of BIST insertion.
- Speed performance of BIST.
- Formal verification.
- Design rule check
- Test coverage.

5.1 VALIDATION

The different validation tasks have been performed for each BIST developed through out the project period. Table 5.1 lists the validation tasks and the tools used during the project work for the above validation tasks. These tools have been operated in UNIX environment through shell commands (Manually) and though UNIX shell scripts (Automatic Procedure).

The development and validation of BIST RTL's have been done for 6 live projects to release the product to the end user on scheduled date. Major development work is done for 65nm high-speed Single-port SRAM memories. Apart from this validation work has been done for Shared BIST, Dual-port RAM and ROM.

Validation Task	**Tool used**	**Comments**
RTL Simulation	SIMVISION from CADANCE	rtl simulation on a given test bench in verilog , vhdl or mixed language.
RTL Synthesis	DESIGN COMPILER From SYNOPSYS	rtl synthesis on user defined constraints
GATE Simulation	SIMVISION from CADANCE	Gate simulation on a given test bench in verilog , vhdl or mixed language.
Formal verification	FORMALITY From SYNOPSYS	Equivalence check between rtl vs. netlist, or netlist vs. netlist with user defined constraints
Rule check	HAL From CADANCE	Based on rule file.
Testability Run	TETRAMAX From SYNOPSYS	Test coverage and fault coverage analysis to check for >98% ,using user defined constraints.

Table 5.1: The Tools used for different validation tasks

5.2 VALIDATIONWIZARD

ValidationWizard is a tool developed by the testsolution team of STMicroelectronics. The objective of this tool is to validate multiple memory BIST designs, netlist or RTL. This tool can automate one or more combination of the above validation tasks.

The tool contains following three components

- Tools: Contains tasks, subtasks, executable files and templates.
- BIN: ValidationWizard executable.
- Configuration files: Template files for design configuration and test configuration.

The two configuration files used by the tool to automate the whole process are –

1. **Design.config file**

   This file contains the following information-

   - Information about design.
   - Search Path
   - Global search path
   - Top design name
   - Testbench
   - Constraint files

2. **test.config file**

   this file contains the following information-

   - Informationabout the task to be performed.
   - Task specific design file
   - Executable files
   - Pass code/failcode

The different features provided by these files are-

- One or more number of validation tasks can be made run or not run with an assertion bit associated with each task.
- We can pass the information about type of input file which is read from the search path. Valid file types supported are – rtl, netlist, model, model, testbench, link library, target library, script.
- The valid languages and formats are – VHDL, verilog , db, txt, ascii .
- We may give the type of output file to be generated after the task which may be used by any subsequent tool in the task list.

- Pass code and fail code can have user defined strings which will be used for validation results.
- This helps validationWizard to generate a complete log report after the execution of the tool.
- There is an option to check the correctness of the configuration file itself.
- Separate directories are generated for each individual task which contains the log files and other generated report.
- After the validation run we can automatically check with the help of a checklist whether all the required tasks have been performed or not.
- Level reports give the detailed information of the Pass/Fail status of the validation task.

5.2.1 Running the ValidationWizard

After making the configuration files the tool is run from the command line following some particular steps.

The tool generates two reports after the execution.

- Level 1 report:
  Like a checklist for all the tasks run and gives a combined information for all the designs mentioned in design.config file.
- Level 2 Report:
  Gives detailed information about each design. It gives the error messages and location for each design and each test case.

5.3 The Shell Scripts

The executable scripts of the tool have been written with shell programming in UNIX environment. The tool is still in the development process with several modifications being made based on the feedback from the team members and other associates. As a part

of this project work the scripts of this tool have been modified for different validation tasks. This modification concerns with changes in the format of the configuration file, the error checking process for the configuration files and the design files etc.

Note : Detailed information can not be revealed here as that is company confidential.

5.4 Validation of BIST for High Speed Single-port SRAM

Validation of BIST RTL for SPHS65 memory has been done with the ValidationWizard tool as well as manually.

5.4.1 SIMULATION & GATE SIMULATION:

The simulation and gate simulation of the design was done with the help of a testbench. This testbench will exercise memory BIST in full *HWDefault* test mode on all memories and for the complete memory address range. It is important to simulate the assembly completely to verify correct operation of the memory BIST.
Different modes of the BIST operation and tests described in chapter 3 have been verified by simulating the entire design (MEMORY+ BIST).

First the simulation of the RTL description of the assembly module. When RTL simulation is successfully completed, the RTL code is synthesized to obtain gate-level descriptions and again the simulation is re-run, which is called as gatesimulation.

The simulation consists –

- Compilation
- Elaboration
- Simulation

5.4.2 SYNTHESIS

Synthesis is done with help of a synthesis script written according to the tool used . Synthesis of the whole design is done based on the constraints for the 65 nm technology. The synthesis consists-

- Read
- Compile

5.4.3 FORMAL VERIFICATION

Formal verification is done to check the equivalence of the RTL description of the design with the Gate level netlist.

5.4.4 TEST COVERAGE

The minimum test coverage should be 98%. Test coverage analysis has been done for the applied test vectors.

5.4.5 HAL RUN

HAL run refers to the process of checking the developed RTL code in reference to the design rules established by the company.

5.5 RESULTS

- Simulation results - Passed
- Synthesis results –
  - Area of MEMORY+ BIST - Approximately 2500 micron
  - Minimum clock period - About 1.4 ns
- Gate Simulation result - PASSED
- Test coverage - Above 98%
- Formal verification - Passed
- HAL Run - Passed

NOTE: The exact results can not be shown as that is company confidential.

CHAPTER 6

CONCLUSION & FUTURE SCOPE OF WORK

The simulation results for the BIST for high-speed 65 nm memories have proved the functional correctness of the BIST design as well as the area and speed performance of the BIST confirms to the expected results.

As more and more memory cores are embedded in state-of-the-art SOCs, embedded memory testing has emerged as a key issue in the VLSI design, implementation and fabrication flow. Low power, low area overhead, low routing congestion, low testing time and reduced performance overhead are a few key issues that need to be tackled.

The Memory BIST has become an integral part of the modern integrated circuits as it provides the best features to test the embedded memories most efficiently with low area overhead and at-speed operation. As the number of memories increases on a chip there is need to develop efficient BIST architectures, which meet the two main criteria's high-speed performance and minimum area overhead.

REFERENCES

[1] A. J. van de Goor. Testing Semiconductor Memories: Theory and Practice. A.J. van de Goor, 1998.

[2] " Essentials of electronic testing for digital, memory and mixed signal VLSI circuits" Michael L. Bushnell & Vishwani D. Agrawal, Kluwar Academic Publishers,2001.

[3] Documents from ST Microelectronics

[4] P.K. Veenstra, F. P. M. Beenker , and J.J. M. Koomen, " Testing of Random Access memories: theory and Practice," *IEE Proceedings G.* vol. 135. no. 1, pp. 24-28 Feb. 1988

[5] A.J. van de Goor and C.A. verruijit, " An overview of deterministic Functional RAM Chip Testing," *ACM Computing surveys,* vol. 22, no. 1, pp. 5-33, Mar. 1990.

[6]P. A. Thakar, Register transfer level fault modeling and test evolution techniques for VLSI circuits. PhD thesis, george Washington university, Washington D. C. , May 2000.

[7] Y. Zorian, E. J. Marinissen, and S. Dey. Testing Embedded-Core Based System Chips. In Proc. IEEE International Test Conference, pages 130–143, 1998.

[8] W. L. Wang, K. J. Lee, and J. F. Wang. An On-Chip March Pattern Generator for Testing Embedded Memory Cores. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 9(5):730–735, October 2001.

[9] P. T. Gonciari, B. M. Al-Hashimi, and N. Nicolici. Test Data Compression: the System Integrator's Perspective. In Proc. of the Design, Automation and Test in Europe Conference, pages 726–731, 2003.

[10] Taiwan Semiconductor Manufacturing Corporation. TSMC Web Site. http://www.tsmc.com.

[11] N. Nicolici and B. M. Al-Hashimi. Power-Constrained Testing of VLSI Circuits. Kluwer Academic Publishers, Frontiers in Electronic Testing (FRET) Series, 2003.

[12]Embedded Memory Bist For Systems-On-A-Chip By Bai Hong Fang, B.Eng. (Electrical) October 2003