Verification Methodologies For Memory BIST

Project Report

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

in

Electronics & Communication Engineering

(VLSI Design)

By

Kunal Parihar

(12MECV13)

Department of Electrical Engineering

Institute of Technology

Nirma University

Ahmedabad-382 481

May 2014

Verification Methodologies For Memory BIST

Major Project Report

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

in

Electronics & Communication Engineering

(VLSI Design)

By

Kunal Parihar

(12MECV13)

Under the Internal Guidance of

Prof. Vaishali Dhare

and

External Guidance of

Mr. Varun Chadha

Department of Electrical Engineering

Institute of Technology

Nirma University

Ahmedabad-382 481

Declaration

This is to certify that

1. The thesis comprises of my original work towards the degree of Master of Technology in VLSI Design at Nirma University and has not been submitted elsewhere for a degree.

2. Due acknowledgement has been made in the text to all other material used.

Kunal Parihar

CERTIFICATE

This is to certify that the Major Project entitled **“”Verification Methodologies for Memory BIST”** submitted by **Mr. Kunal Parihar (12MECV13)**, towards the partial fulfillment of the requirements for the degree of **Master of Technology in VLSI Design** of **Nirma University of Science and Technology, Ahmedabad** is the record of work carried out by him under our supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for the examination. The results embodied in this major project, to the best of our knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

External Guide	**Internal Guide**
-------------------	-------------------
Mr. Varun Chadha	Prof. Vaishali Dhare
Technical Lead	Assistant Professor
Memory Test and Soft IP's(MTS), TR&D	Institute of Technology
STMicroelectronics India Pvt. Ltd.	Nirma University
PG Coordinator VLSI Design	**HOD**
-------------------	-------------------
Dr. N. M. Devashrayee	Dr. P. N. Tekwani
Professor	Professor
Institute of Technology, NU	Institute of Technology, NU
Director	

Dr. K Kotecha	
Director, IT-NU	
Date: ___________________	**Place: Ahmedabad**

Acknowledgment

I would have never succeeded in completing my Thesis without the cooperation, Encouragement and help provided to me by various people.

Firstly, my sincere thanks to the MTS team, especially the Verification team, for their help during this training. Their wisdom, clarity of thought and support motivated me to bring this project to its present state.

I am highly indebted to my Project manager, **Mr. Nitin Sharma**, my immediate supervisor **Mr. Varun Chadha** and also to **Mr. Nitin Ahuja** for providing necessary information regarding the project and also for their constant guidance, supervision, kind co-operation, and invaluable support in all aspects. My thanks and appreciations also go to my colleagues and team members in developing the project and for providing me with a lively and energetic work environment.

I would like to express my sincere gratitude to **Dr. Ketan Kotecha** (Director, Nirma University, Ahmedabad) for his continuous guidance, support and enthusiasm. I would take this opportunity to thank **Dr. P. N. Tekwani** (Head of Department, Electrical Engineering), **Dr. N. M. Devashrayee** (Professor and Program Coordinator, M.Tech - EC (VLSI Design)), Internal Guide **Prof. Vaishali Dhare** (Assistant Professor, M.Tech - EC (VLSI Design)) and all the faculties at **Nirma University (VLSI Design)**, for their vision and relentless effort, support, and encouragement to provide me with this excellent opportunity to carry out my project work in such a highly renowned and esteemed organization, ST Microelectronics. I am equally thankful to **ST Microelectronics** for providing me the invaluable exposure to the industry and the current market trends.

Finally, I would like to express my heartfelt thanks to my parents and colleagues for their blessings and for their constant love and support.

Kunal Parihar
(12MECV13)

Abstract

The Project deals with functional verification of the MEMORY BIST and supported collar models and controller with an intent to verify all possible modes, application of all the tests by the BIST to ensure desired functionality. Our verification team is responsible for verification of collar and controller models of BIST for all types of memories such as SP-SRAM (Single Port SRAM), DP-RAM (Dual Port SRAM), ROM and BIST Controller as well as all types of supported architectures such as C28FDSOI and M40 NVM architectures. The aim of verification is to establish a common to all and stable verification environment /test bench which can serve to verify all possible configurations of Memory BIST collar and controller. The verification methodologies aim to provide a robust solution to testing and verifying the Memory BIST IP for its given specification covering maximum possible corner cases to deliver an efficient and completely working design.

ST Microelectronics At A Glance

- A world leader in providing the semiconductor solutions that help our customers improve quality of life for everyone, both today and in the future
- Among the world's largest semiconductor companies
- A leading Integrated Device Manufacturer serving all electronics segments
- A leading technology innovator (around 12,000 researchers approx. 21,500 patents)
- Key strengths in Multimedia Convergence, Power Applications and Sensors
- Rich, balanced portfolio (ASICs, Application-Specific Standard Products and Multi-Segment Products)
- A pioneer and visionary leader in sustainability
- President and CEO: Carlo Bozotti
- 2011 revenue $9.73 billion
- Approximately 50,000 employees including STEricssion at December 31, 2011
- Advanced research and development centers in 10 countries
- 12 main manufacturing sites
- Corporate Headquarters Geneva, Switzerland

- Global presence with sales offices all around the world
- Public since 1994 - shares traded on New York Stock Exchange (NYSE: STM), Euronext Paris, and Borsa Italiana
- Created as SGS-THOMSON Microelectronics in June 1987, from merger of SGS Microelettronica (Italy) and Thomson Semiconducteurs (France)
- Renamed ST Microelectronics in May 1998

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction to BIST

1.1.1 Description

Built-in self-test (BIST) is a set of structured-test techniques for combinational and sequential logic, memories, multipliers, and other embedded logic blocks. In each case the principle is to generate test vectors, apply them to the circuit under test (CUT) or device under test (DUT), and then check the response. BIST - part of the circuit (chip, board or system) Is used to test the circuit itself. In other words BIST is defined as a DFT technique in which testing (test generation and test application) is accomplished through built-in hardware features.

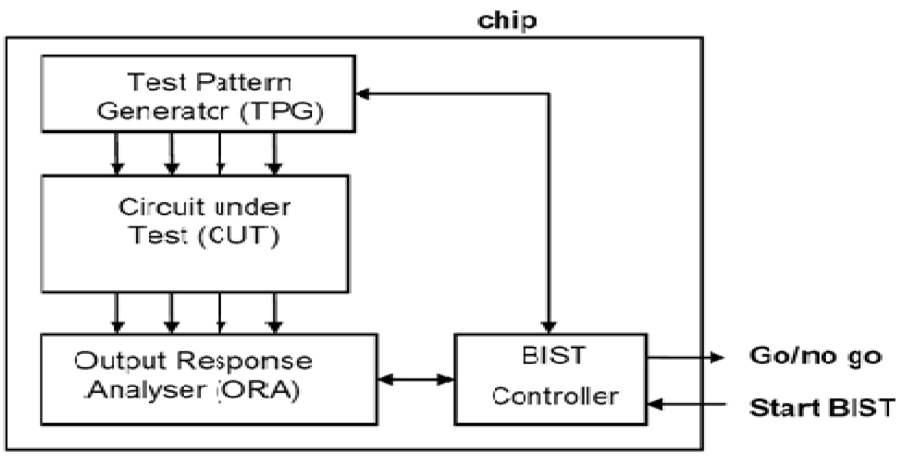


Figure 1.1: A Typical Logic BIST System

Essential circuit modules required for BIST include:

- BIST Controller
- Test Pattern Generator(TPG)
- CUT(Circuit Under Test)
- ORA(output response analyzer)

1.1.2 BIST Controller

- Inhibit system clocks and control test clocks.
- Communicate with other test controllers using test buses.
- Control self test operations and Seeding of registers.
- Keep track of the number of shift commands required in a scan operation.
- Keep track of the number of test patterns processed.

1.1.3 TPG(Test pattern generator)

- A Pseudo Random Binary Sequence (PRBS) generator allows us to generate all of the required binary patterns for the circuit under test.
- It is random in a sense that the value of an element of the sequence is independent of the values of any of the other elements.
- It is 'pseudo' because it is deterministic and after N elements it starts to repeat itself, unlike real random sequences.
- Basically PRBS is implemented using LFSR (Linear Feedback Shift Register).
- LFSR is an n-bit shift register which pseudo-randomly scrolls between values, but does it very quickly because there is minimal combinational logic involved.
- Once it reaches its final state, it will traverse the sequence exactly as before.

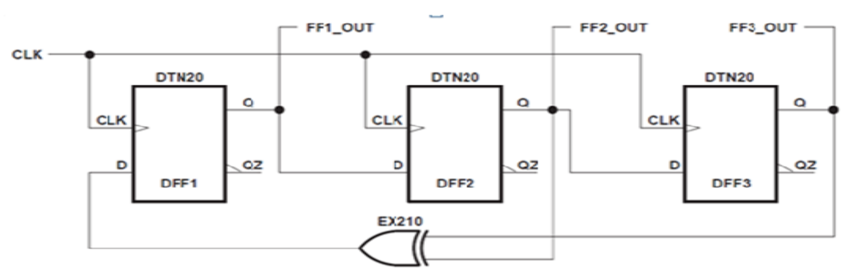


Figure 1.2: Conventional LFSR

1.1.4 ORA(Output response analyzer)

ORA are built-in logic block observer register (BILBO). The circuit is shown in figure below which allows four different modes controlled by C0 and C1.

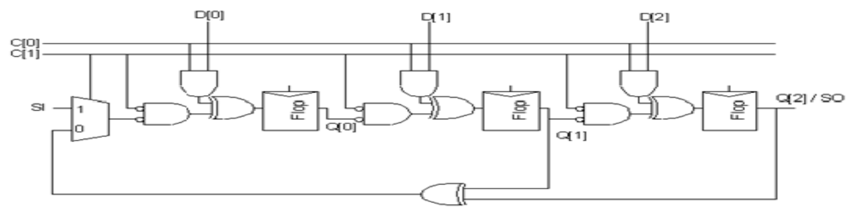


Figure 1.3: Bit built-in logic block observer register (BILBO)

1.1.5 BIST Benefits:

Faults Tested:

- Single combinational / sequential stuck-at faults.
- Delay faults.
- Single stuck-at faults in BIST hardware.

Advantages of implementing BIST include:

- Reduced testing and maintenance cost.
- Lower test generation cost.
- Reduced storage / maintenance of test patterns.
- Simpler and less expensive ATE.
- Can test many units in parallel.
- Shorter test application times.
- Can test at functional system speed.
- Better fault coverage.
- Easier customer support.
- Capability to perform tests outside the production electrical testing environment.

Disadvantages of implementing BIST include:

- Additional silicon area and fabrication processing requirements for the BIST circuits.
- Reduced access times.
- Additional pin (and possibly bigger package size) requirements, since the BIST circuitry need a way to interface with the outside world to be effective.
- Possible issues with the correctness of BIST results, since the on-chip testing hardware itself can fail.

1.1.6 Test Description:

With the ever increasing complexity and gate counts of modern devices, a number of testability problems have been encountered. One of the fundamental issues is the complexity and size of the test program required to test these devices. The test program is a necessary way to ensure that the high quality standards demanded by the market are met. In brief, two factors are in favor of smaller test programs. Firstly the smaller the program the fewer the number of test vectors and therefore the faster it can be run. Thus, test time can be reduced.

Secondly if the program is small, problems with the available memory on testing machines can be avoided. One of the main factors which greatly increases the number of test vectors required to test a device is the use of large embedded memories. Hence, if these memories can test themselves a great reduction in test program size can be achieved.

A BIST block is an off-line verification of the Circuit Under Test (it implements the algorithm to test it), as opposed to an on-line verification or concurrent test. This test can execute at the same time the BIST of all on-chip memories and report any failure using BBAD signals. This test checks all the memories cuts inside the chip through the BIST architecture. For various single and dual port memories if we use the clock running at 72MHz, in that case the time delay in all these memories is more than 1 clock period, so in that case there would not be any synchronization available at the top. To avoid this we slow down the clock by a factor of 3 and this clock should be activated after getting all MEM bends.

Key Feature / Objectives

- Modular BIST system
- Optimized for sharing approach
- Provides flexibility to ease usage at SoC level
- Provides debug, diagnostic and repair capabilities
- Support for user defined test patterns

1.2 Overview of Memory BIST features

Memory BIST

A BIST is an added block to the memory for off-line verification of Circuit under Test (it implements an Algorithm to test it) as opposed to on-line verification or concurrent test.

MEMORY + BIST

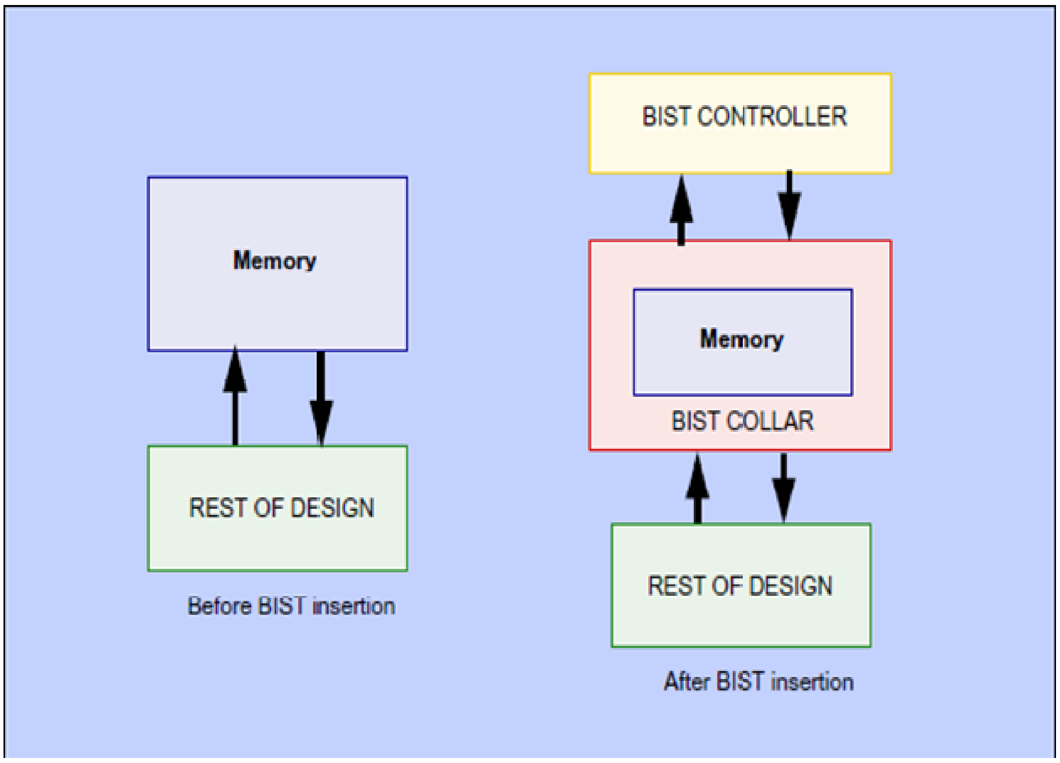


Figure 1.4: showing memory with BIST

Features of BIST Collar

March Element Register:

A March element is a sequence of operations performed by BIST on the memory based on a criterion. This set of registers is used to store the algorithm to be executed by the collar. Number of bits required to store algorithms is constant for a given memory type.

Serial Interface Registers:

These registers load and update various information to and from the collar. These registers are present at collar to controller serial interface.

Types of Serial Interface Registers:

- March element register / Repair register
- Status register
- Execution status register
- Diagnosis register
- Signature register
- Collar select register
- Resource bad register
- Bypass register

Repair Registers:

Repair registers of BIST store the repair information for each memory. These registers provide repair information about Fault locations, type of redundancy and availability of repair resources and their status.

Programmability Feature:

This feature provides direct interaction between collar and user through dedicated link between collar and controller. The user can directly program the march element registers. It is useful because default algorithms not sufficient to find out all types of faults so need for programming by the user. The programming of all collars simultaneously is possible.

DIAGNOSIS Feature:

It is an interactive feature of BIST which helps in diagnosis for user. It helps in diagnosis of faults detected by BIST in memory. Every time a fault occurs , the Diagnosis information is shifted out to the user before the next march operation run. Easy interface with tester to create complete fault map of memory.
Diagnosis register contains following information:

- Root cause of faults
- Address at which fault occurred
- Data at the corresponding address
- FSM states of BIST Run
- March operation and operation count

Chapter 2

Types Of Faults And Faults Targeted By BIST

2.1 Fault in the Memories

FAULT MODELS:

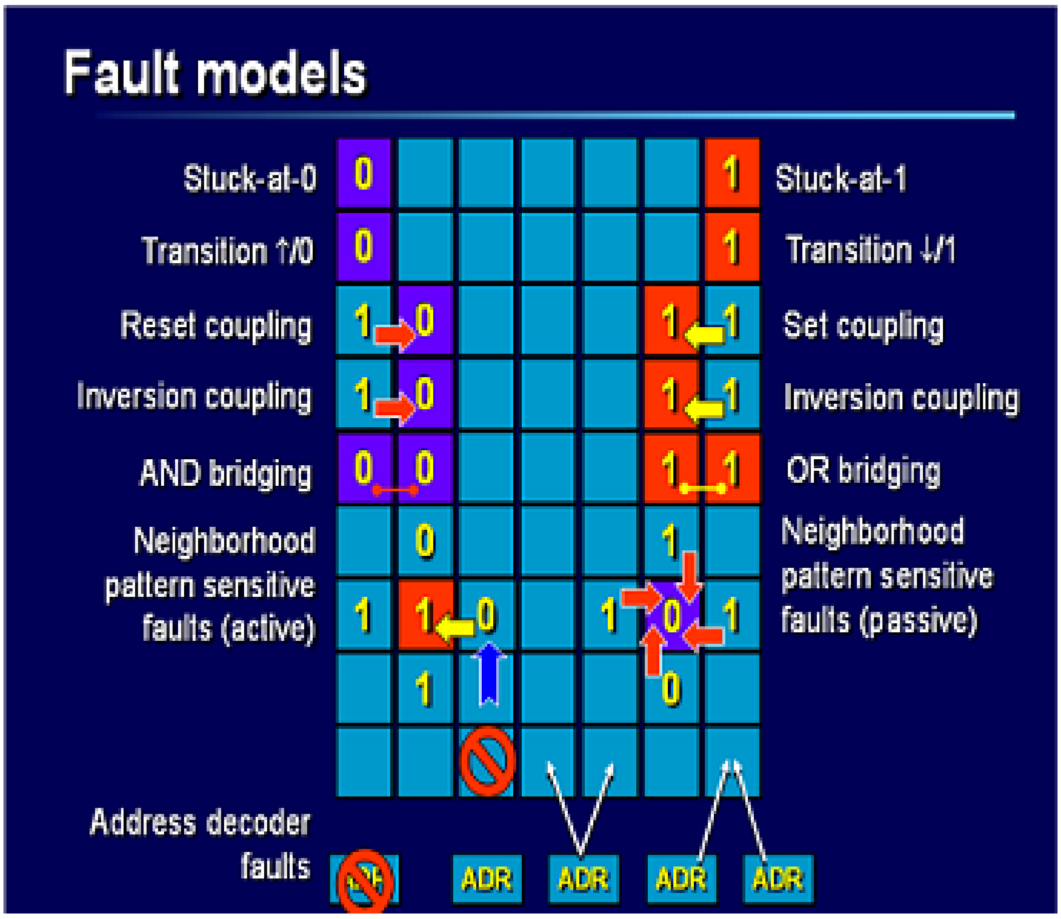


Figure 2.1: Fault model types

Fault Types:

- **Permanent:** System is broken and stays broken the same way indefinitely
- **Transient:** Fault temporarily affects the system behavior, and then the system averts to the good machine - time dependency, caused by environmental condition
- **Intermittent:** Sometimes causes a failure, sometimes does Not.

Failure Mechanisms

- Permanent faults:
- Missing/Added Electrical Connection
- Broken Component (IC mask defect or silicon-to metal connection)
- Burnt-out Chip Wire
- Corroded connection between chip & package
- Chip logic error (Pentium division bug OR Pentium FDIV bug)

Transient Faults:

- Cosmic Ray
- An particle (ionized Helium atom)
- Air pollution (causes wire short/open)
- Humidity (temporary short)
- Temperature (temporary logic error)
- Pressure (temporary wire open/short)
- Vibration (temporary wire open)
- Power Supply Fluctuation (logic error)
- Electromagnetic Interference (coupling)
- Static Electrical Discharge (change state)
- Ground Loop (misinterpreted logic value)

Intermittent Faults:

- Loose Connections
- Aging Components (changed logic delays)
- Hazards and Races in critical timing paths (bad design)
- Resistor, Capacitor, Inductor variances (timing faults)
- Physical Irregularities (narrow wire high resistance)
- Electrical Noise (memory state changes)

Fault Modeling:

- Behavioral (black-box) Model:
- State machine modeling all memory content combinations: Intractable
- Functional (gray-box) Model:used
- Logic Gate Model: Not used - Inadequately models transistors & capacitors
- Electrical Model: Very expensive
- Geometrical Model
- Layout Model: Used with Inductive Fault Analysis

Reduced Functional Faults

SAF	Stuck-at fault
TF	Transition fault
CF	Coupling fault
NPSF	Neighborhood Pattern Sensitive fault

Different Fault Types:

Stuck-at Faults:

The memory is said Stuck-at faults (stuck-at 0 and stuck-at 1) when the logic value of cell or line always at state 0 or 1, cannot be changed to opposite state. These faults imply that either a cell or line is struck to logical 1 or 0 which is called struck-at-1 or struck-at-0 respectively.

Condition: For each cell, must read a 0 and a 1.

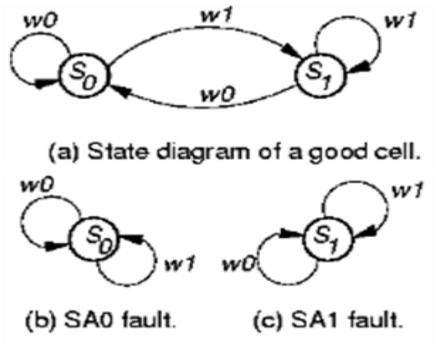


Figure 2.2: Stuck-at-fault

Transition Faults:

Transition faults been similar to struck-at-faults, however in this case once the memory cells are written to one state is not impossible to transition back. - Cell fails to make 0 1 or 1 0 transitions Condition: Each cell must undergo a transition and a transition, and be read after such, before undergoing any further transitions.

Coupling Faults:

Coupling faults involve coupling between two adjacent cells when a 0 to 1(or 1 to 0) is written to a cell, it caused the neighbor cell to change its desired value. These faults can be classified into inversion or idempotent.

- Coupling Fault (CF):
- Transition in bit j causes unwanted change in bit i
- 2-Coupling Fault:

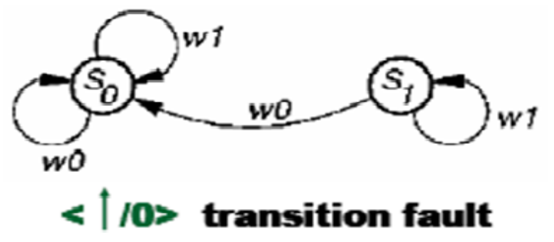


Figure 2.3: Transition faults

- Involves 2 cells, special case of k-Coupling Fault
- Must restrict k cells to make practical
- Inversion and Idempotent CFs:
- special cases of 2-Coupling Faults
- Bridging and State Coupling Faults involve any # of cells, caused by logic level
- Dynamic Coupling Fault (CFdyn):
- Read or write on j forces i to 0 or 1

Neighborhood pattern Sensitive faults (NPSF):

These faults involve a base memory cell in the centre which surrounded by eight neighboring cells. It is possible for base memory cell to transit to a certain memory cell to transit to a certain value influenced by its neighborhood cells.
A cell or line which fails to undergo a 01 transition when it is written is said to contain an up transition fault; similarly, a down transition fault is the inability to make a 10 transition. A test that has to detect and locate all transition faults (TF) should satisfy the Following requirement:

Each cell must undergo a transition (state of the cell goes from 0 to 1), and a - transition (state of the cell goes from 1 to 0), and be read after each transition before undergoing any further transitions.

Data Retention faults (DF):

These faults occur in memories that are unable to retain their value some time after a write or read operation.

In general, DRFs are supposed to be produced by very high resistive-open defects that affect the refreshment loop of the core-cell. We demonstrate that lower values of resistance may produce hard to detect DRFs. Moreover, each resistive open defect produces a particular faulty behavior of the core-cell that changes for different ranges of the resistive value. We analyze different cases and we propose for each one an efficient test procedure based on March tests. In particular, we propose to stimulate the defective cells in some cases by indirect accesses and in some other cases by emphasizing natural noise phenomenon of SRAM memories.

A DRF occurs when a memory cell loses its previously Stored logic value after a certain period of time during which it has not been accessed In general this kind of fault is the consequence of resistive-open defects in SRAM core-cells, in particular in the self-refreshment loop circuit. Figure 1 depicts the scheme of a standard six transistors core cell in which we have inserted three resistive-open defects in locations where they are commonly the cause of DRFs .

. These defects are denoted as Df1, Df2 and Df3.

Address Decoder Faults (AF):

These faults occur due to no cell can be accessed with a certain address multiple cells are accessed simultaneously or a certain cells can be accessed with multiple addresses.

A two-port memory contains two duplicated sets of address decoders, which operate independently. Testing such memories requires the use of single-port tests as well as special two-port tests semi, the test strategy determines which tests have to be used. Many two-port memories have ports which are read-only or write-only

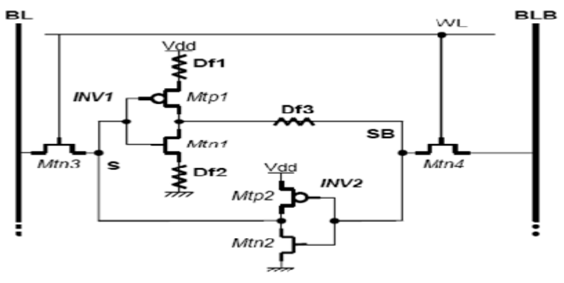
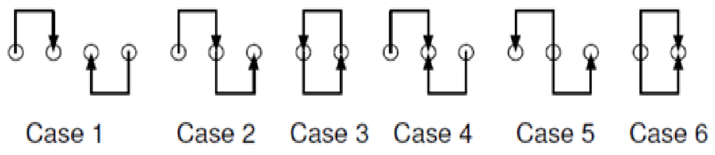


Figure 2.4: Resistive-open defects injected into the memory core-cell as cause of DRFs

semi; this impacts the possible tests for single-port and two-port memories, as well as the test strategy. In this paper the effects of interference and shorts between the address decoders of the two ports on the fault modeling are investigated. Fault models and their tests are introduced. In addition, the consequences of the port restrictions (read-only or write-only ports) on the fault models and tests are discussed, together with the test strategy.

Fault Combinations:

The most relevant functional fault models and, at the end, we have outlined that in address decoders faults are interrelated. However, when testing a memory core, it is very likely that many various types of faults may occur simultaneously.



These faults can be linked or unlinked. In a linked fault one fault may influence the behavior of other faults. An unlinked fault does not influence the behavior of other faults. Linked faults can be further classified as linked with the same fault type or linked with different fault types.

Linked Faults of the Same Fault Type

Since a SAF involves only one cell and only one SAF can occur in a single cell, a SAF cannot be linked with another SAF. Similarly, a TF also cannot be linked with another TF. In a CF, two cells are involved. Figure 2.4 lists 6 different cases when two coupling faults (4 cells involved) occur concurrently. Case 1,2,3 and 5 are unlinked faults because each of the coupled cells are only coupled in only one way. Case 4 is a linked fault because a cell is coupled to more than one cell and case 6 is also a linked fault because a cell is coupled to a single cell in more than one way. As stated in, in general, linked CFins cannot be detected by March tests. However, tests for idempotent CFs (CFids) will detect inversion CFs (CFins).

Linked Faults of Different Fault Types

When a test for a certain fault type is performed, it will cover faults at a lower hierarchical level (see Table 2.2). When SAFs link with TFs and/or CFs, they can be detected without any extra tests for TFs and/or CFs. For unlinked TFs and CFs, testing CFs can also detect TFs. However, if TFs are linked with CFs then require a new test. This is because a TF may mask a CF, while the CF masks the TF. For a full description of linked faults.

Functional Testing and March Test Algorithms

Based on the used memory fault models, memory test algorithms can be divided into four categories [39] as described below:

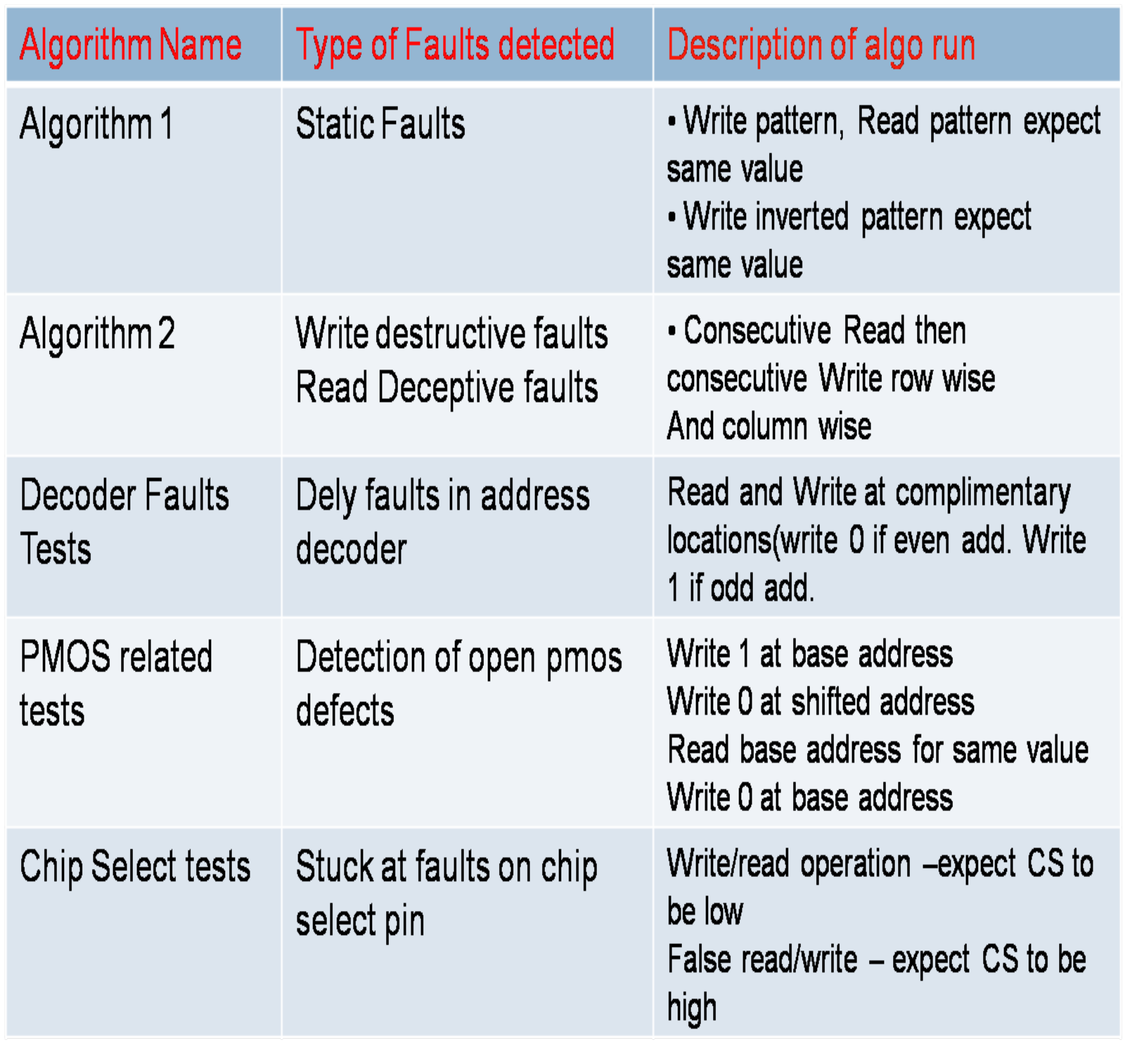
1. Traditional tests including Zero-One, Check board, GALPAT and Walking 1/0, Sliding Diagonal and Butterfly. They are not based on any particular functional fault models and over time have been replaced by improved test algorithms, which result in higher fault coverage and equal or shorter test time.

2. Tests for stuck-at, transition, and coupling faults those are based on the reduced functional fault model and are called March test algorithms.

3. Tests for neighbourhood pattern sensitive faults.

4. Other memory tests: any tests which are not based on the functional fault model are grouped in this category. March test algorithms can efficiently test embedded memories and, therefore, the rest of this section provides more details about them.

2.2 Various BIST Algorithms and Faults Targeted



Algorithm Name	Type of Faults detected	Description of algo run
Algorithm 1	Static Faults	• Write pattern, Read pattern expect same value • Write inverted pattern expect same value
Algorithm 2	Write destructive faults Read Deceptive faults	• Consecutive Read then consecutive Write row wise And column wise
Decoder Faults Tests	Dely faults in address decoder	Read and Write at complimentary locations(write 0 if even add. Write 1 if odd add.
PMOS related tests	Detection of open pmos defects	Write 1 at base address Write 0 at shifted address Read base address for same value Write 0 at base address
Chip Select tests	Stuck at faults on chip select pin	Write/read operation –expect CS to be low False read/write – expect CS to be high

Figure 2.5: Algorithms for various types of faults

Chapter 3

Introduction to Verification Methodologies

3.1 Need for functional verification:

The primary purpose of functional verification is to detect failures so that bugs can be identified and corrected before it gets shipped to customer. This builds confidence among customers and helps the organization to stay in business. Functional verification is needed to verify the correctness of RTL design as well as on broader aspects to verify the correctness of Specification Implementation and see if any specification misinterpretation has occurred. Functional Verification is a N-P hard problem since it involves sheer volume of test cases even for simplest of designs.

Main components of functional verification:

1. Verification Environment
2. Functional Coverage
3. Code Coverage

Verification Environment:

It usually consists of a layered test bench and its various components like stimuli/generator , driver ,checker ,monitor which apply stimuli to the DUT based on test scenarios defined and then expect results to be compared with expected behavior .

Functional Coverage:

It defines the coverage points based on the functional behavior of design as per the specification. It is an important tool to identify the correctness of design specification implementation and completeness of RTL code.

Code Coverage:

It reflects how thoroughly the source HDL code is exercised. The various criteria for code coverage include statement coverage / line coverage block coverage, toggle coverage, fsm coverage branch coverage and path coverage.

3.2 Layered Test Bench Architecture:

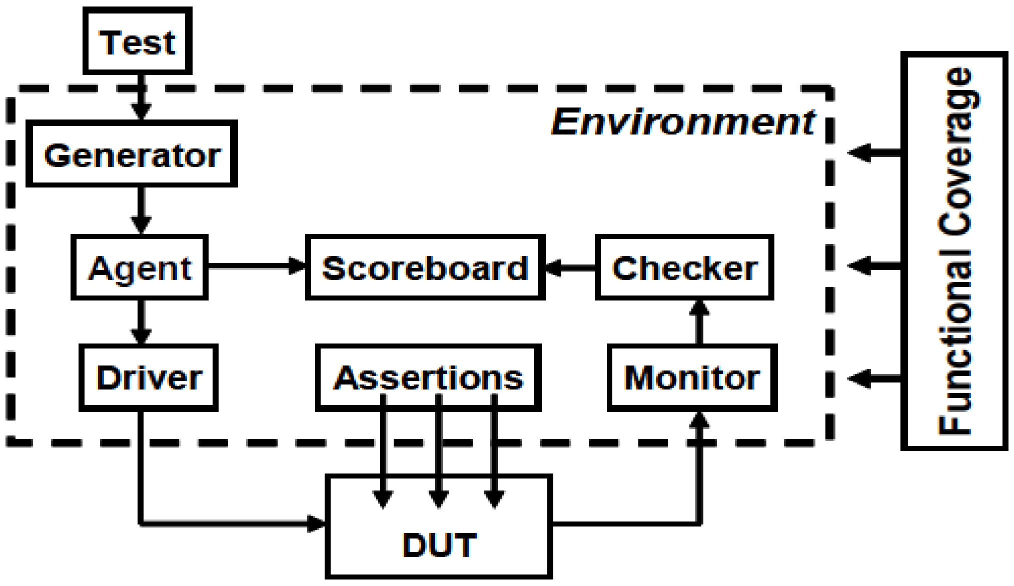


Figure 3.1: Layered Test bench

Components of a layered test bench:

1. **<u>Generator:</u>** It generates the stimuli for the DUT (Design under Test). The stimulus consists of all the tests that are to be run on the DUT.

2. **<u>Agent:</u>** It acts a transactor which takes higher level transactions like DMA operation from the generator / test class and converts it to signal level to be given to the driver

3. **<u>Driver:</u>** It takes the stimuli provided by higher layers of test bench and converts them into actual inputs for design under verification.It forms a part of the signal layer of a layered test bench and drives signals like chip select/ write/ read directly on the DUT .It also drives reset pins if and when a reset sequence is called.

4. **<u>DUT:</u>** It consists of the Design Under test. It may be a complete IP or a part of IP or any design that needs to be verified for its functionality.

5. **Monitor:** It converts the state of the design and its outputs to a transaction abstraction level so it can be stored in a 'score-boards' database to be checked later on. It observes and passes out the values of actual signals from DUT to higher abstraction level layers.

6. **Scoreboard:** It takes the various output signals from the Design under Test / Verification (DUT / DUV) and compares them with the expected behaviour and thereby declares a bug in design .It contains various tasks for expected golden reference which is to be compared to incoming actual information from DUT.

7. **Checker:** Checker converts the low level data to high level data and validated the data. This operation of converting low level data to high level data is called Unpacking which is reverse of packing operation. For example, if data is collected from all the commands of the burst operation and then the data is converted in to raw data, and all the sub fields information are extracted from the data and compared against the expected values. The comparison state is sent to scoreboard.

8. **Assertions:** They are a set of expected behaviour for a set of signals at the occurrence of some specific event / clock .They act as constraints on a design behaviour obtained from design specification and asserted every time a design is to be checked for functional verification. They are independently applied to the DUT and separated from the directional and random checking by other components of test bench

3.2.1 Memory BIST Specific Test Bench Architecture:

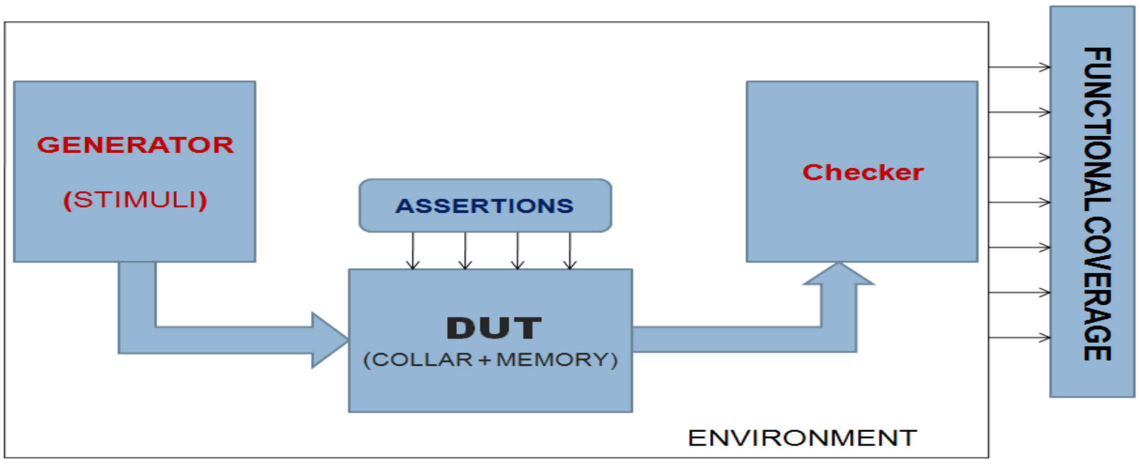


Figure 3.2: Verification Environment for M-BIST

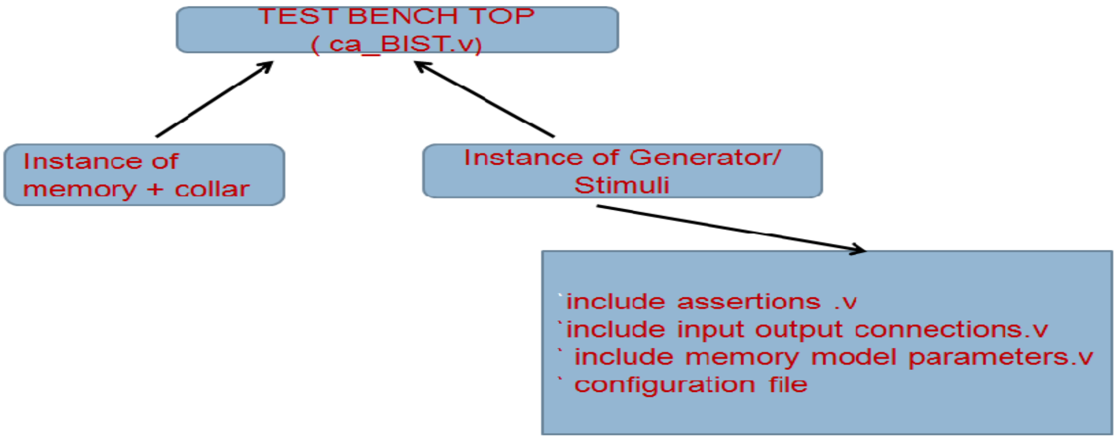


Figure 3.3: TEST BENCH FLOW FOR COLLAR

Components of a Functional Testbench

1. **Generator:** It generates the stimuli for the DUT (Design Under Test). The stimulus consists of all the tests that are to be run on the DUT.

2. **DUT:** The Design Under Test is the circuit/unit to be verified .In case of MBIST there may be two configurations:

   - Collar + Memory verification of collar for BIST
   - Controller verification of controller for BIST

3. **Checker:** It serves the purpose of both checker and scoreboard and compares the various test outputs with expected outputs.

4. **Assertions:** gives common assertions for all signal pins of BIST (E.g. : March End , CSN,TBIST etc) and Test scenario assertions (generator specific assertions)

5. **Input output connections:** all connectivity signals

6. **Instance of stimuli:** this file does not have a module only end module.

7. **Instance of memory + collar (collar verification):** for memory and collar related pins.

8. **Configuration file:** consists of feature parameters (Bitmap, repair register, redundancy etc) and architecture parameters (mux size, no of banks, add size and data size etc).

Steps for verification flow:

1. Prepare a Test Plan /modify a test plan based on specification
2. Modify the existing Environment according to above test plan
3. Do Random Testing
4. Wherever coverage is less include directed test cases
5. Include predefined Assertions
6. Check Functional Coverage
7. Run regressions in all three mode :
    - Functional RTL verification
    - Functional Netlist Simulation
    - Timing Simulation on Netlist

3.3 Methodologies adopted for MBIST verification for different memory architecture.

1. **The Verification / Test Plan:**
    - Document prepared by verification team identifying all the scenarios to be tested as per functional specifications
    - All the tests to be run on BIST identified (Directed Testing)
    - Random scenarios identified for each test (Randomized Testing)
    - Test of specific features and the modes of BIST each test should cover also identified (Redundancy Test, BITMAP test, Default algorithm tests with and without fault insertion).
2. **Grey Box Verification Approach:**
    - While verifying the collar model the controller is treated as a black box only providing the March elements and sending signals to the collar.
    - Only the collar is verified for its complete functionality.
    - While verifying the controller the collars are treated as black box and the transactions of controller are only verified.

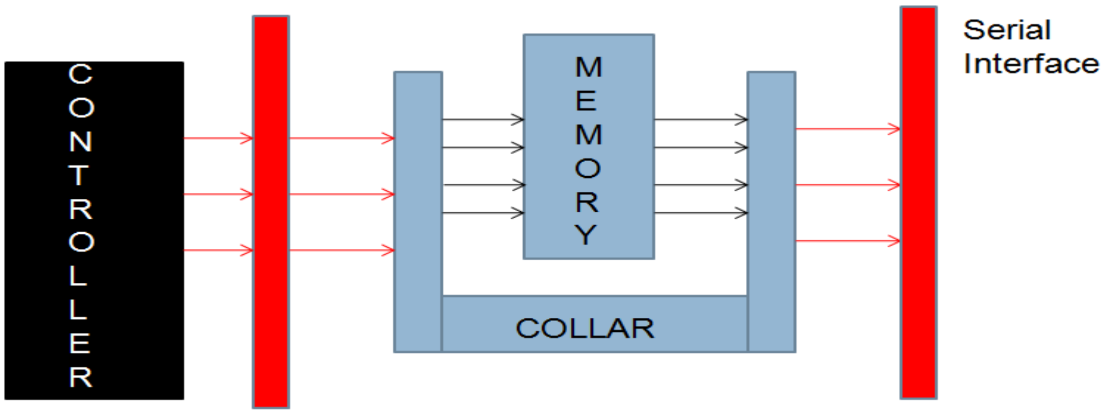


Figure 3.4: Collar + Controller interface

3. **Random Testing**

   - Random testing by generating random scenarios
   - Random programming of march element registers
   - Random testing at signal level (Cycle based test bench) i.e. adding extra shift cycles, shifting out BITMAP data at different clock edges.
   - Randomly asserting different modes and flags in the test bench
   - Negative testing
   - Random clock generation (clock frequency changed every 100 cycles)
   - Random Seed Based testing (Time Seed used as reference)
   - Verilog constructs such as $random, $dist_uniform used to select and deselect various modes of BIST.

4. **Assertions and Functional Coverage**

   - Test Scenario Assertions:
     - No bad status when no faults inserted on memory
     - Chip select signal in stable mode
     - All signal and mode values expected during various test scenarios
   - Common Assertions:
     - Expected value on input / output pins
     - March end assertions (signal for end of march operation)
     - TBIST assertions

5. **Gate level Simulation and Timing Simulation**

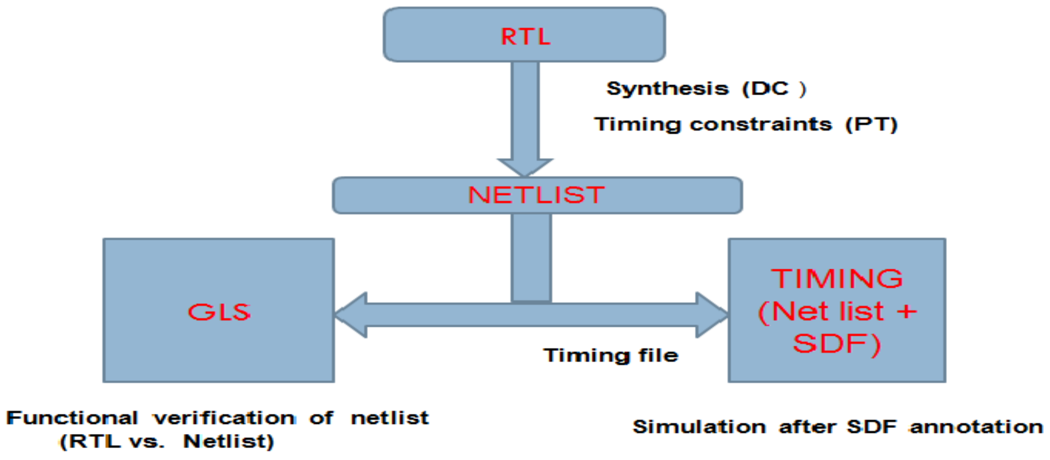


Figure 3.5: Flow of GLS and Timing Simulation

GATE LEVEL SIMULATION:

- Done on functional netlist with zero delays (only inertial delays)
- To functionally verify design at gate level
- Specific tests to run only during GLS (Scan collar test)

TIMING SIMULATION:

- adding actual delays to design
- constraints such as Multicycle paths, False path added
- SDF(Standard delay format) back annotation done
- Timing file provided by designer
- Specific blocks in test bench written only to be run in timing simulation
- Synchronization cycles adjusted according to timing simulation.

Chapter 4

Project Work

4.1 Verification of C28SOI Collar

- **Modification of Test plan:**
  - New test features added for changed technology.
  - Test strategy defined
- **Changes in implementation of Redundancy test:**
  - Repair information registers present inside the BIST
  - So test to check repair information generated and values of corresponding serial interface registers
  - Status check after repair information calculation
  - March element register by purpose
- **Invalid Test support for Single Bank Memory:**
  - For memories with configuration of bank, bank decoder test will not run.
  - For such memories the March operation must be stopped and signal march end should be high.
  - Programming the status of BIST to avoid false status.
- **Invalid Test for single mux memory:**
  - For mux =1 ,memories will have single column so address counter cannot run in column fast mode i.e unable to switch addresses column wise.
  - Status of all MARCH ELEMENT run for tests with column fast address counter must be changed.
- **Gate level and Timing simulation:**
  - Zero delay simulation for the generated netlist.
  - Design Compiler tool from Synopsis used to generate Netlist
  - Prime Time tool from Synopsis for timing analysis.
  - Standard cells libraries and Memory (cells) libraries provided in synthesis setup file

  - Operating conditions (Typical, MAX, MIN) supplied
  - Verified for worst case PVTs (process, voltage, Temperature).
  - Constraints of multicycle paths for ATPG pins and false paths for synchronization flops .
  - SDF back annotation using these exceptions and constraints.
  - Timing file included to provide false paths for which timing violations to be ignored

- **Run Regressions on Different configurations:**
  - Preparing a user configuration file with different criteria
  - Largest(max words ,max bits, max muxes and max bank), Smallest, Widest and Shortest criteria identified according to memory specification
  - Criteria of dedicated, parallel shared and sequentially shared collars.
  - Combination of various BIST and memory options selected
  - Cuts with and without BITMAP, dedicated clock tree and Redundancy selected.
  - Different types of regression runs in all three modes (Functional ,Gate level and Timing Simulation):
    - Parallel run
    - Single cut run (for debugging)
    - Single cut multiple run (time seed based)
    - Run of faulty cuts
    - Gatesim run
    - Timing run

4.1.1 L1Cache Memories

- New default tests added in Test bench For C28 platform, Default tests with fault insertion have been added.
- Test for Default BITMAP Run added Diagnosis test added using default algorithm of BIST.
- Invalid test support Bank decoder test cannot be run in Single bank case.
- Pipelined BIST support
  - L1 cache memories are having 2 pipeline cycles at the input side and 1 pipeline cycle at the output side]item To compensate for these BIST has to be aligned with respect to memories.
  - Synchronization cycles between controller clock /tester clock and
  - Memory clock adjusted to current test bench. (1 Tester clk 3 Sync cycles 3 memory clk)
- Changes corresponding ATPG Mux support in M-BIST
  - Memory initialization pin (INITN) controlled by mux in new TECH
  - Improved controllability at collar level
  - Support in test bench provided
  - Connectivity of input output ports modified.
- Bug identified and test bench/ simulator races resolved
  - Error in BITMAP test
  - Identified signal mismatch of RTL.
  - Races in simulator used NCSIM from Cadence
  - Problem due to specific Verilog constructs
  - Resolved using alternatives.

4.1.2 Ternary CAM (TCAM) Memories

- **Change in Address Generation and Data Generation**

  Modified checker for change in expected address and data for various march elements.

- **Change in March elements**

  Data background bits changed so algorithm behaviour changed

- **Change in Diagnosis Test**

  Information shifted out after fault diagnosis delayed by one cycle.

  The corresponding expected register fields required to be changed.

- **Change in Latency for Read and Write operations**

  Cycle Based Test Bench is completely re synchronized to accommodate the read cycle latency which changes from 5 to 13 adding extra wait cycles and corresponding shift in cycles for checking.

- **New March Element fields added**

  This is a major change since the search algorithms have changed completely with respect to new fields in march element register.

  The XY_en field is now used to determine the search address and data corresponding different scenarios.

  Since hit enable test now removed the pins corresponding it i.e Mhto are removed and replaced by pins to identify address decoding. i.e hf_add.

4.2 Verification of NVM M40 Collar

- Support for new options of memories
- Support for new redundancy schemes
- Defining Test strategy for M40 architecture based MBIST collar

- Run the Regressions for gate level simulations and Timing simulation
- Support for new options of memories
  - Many default memory options made optional
  - So these options implemented from command line (if command line contains these options they are added in Collar TB parameters file.
  - New memories have different command line options for different I/O pins.
  - Test bench to support these signal pins and expected values
- Support for new redundancy schemes:
  - Support of same BIST for all types of M40 memories offered
  - Some memories with column type redundancy.
  - Some memories with row type and column type redundancy.
  - So test bench being modified for both types of redundancy schemes
- Unresolved signal values in memory model :
  - A particular value in provided memory model tied to X.
  - The corresponding BIST signal pins also becoming x.
  - Only occurs when input pin of memory (STOV) goes high in one particular mode
  - Memory still functionable.
  - So test bench to be isolated from the changes for expected behaviour
  - The signals of test bench modified
  - A tri reg is defined by multiplexing the signal with logic Z.
  - Test bench is checked on prototype of various sizes and declared mature.

4.3 Verification of BIST Controller

- Modification in Verilog Environment of BIST Controller
  - Change in features like march element register width and and change in redundancy scheme impact the controller and hence the verification model needed to be updated
- Modification of Collar Model used as Bus Functional Model for the controller verification
  - The dummy collar model is used to replicate collar behaviour via serial interface between collar and controller and it needs to be updated as per changes march algorithms and redundancy schemes.
- Development of System Verilog Environment for C32 architecture controller:
  - With Verilog model of verification environment as reference and the need for extensive and higher levels of abstraction and improvement in coverage a new model of verification environment in system verilog object oriented programming language is developed.
- Study of UVM Methodology based Environment of Controller
  - In order to have a stable yet exhaustive test bench structure a Universal methodology based verification environment has been developed and being used to verify the Controller of Memory BIST sub systems .It has an extensive coverage based and assertion supported test bench that interacts with the DUT through various transactions and signals driving layers. Study and understanding of this environment was used to accomplish verification demands for continuously changing architectures and features of both memory and BIST and validating those changes for optimum and functionally correct behaviour.

Chapter 5

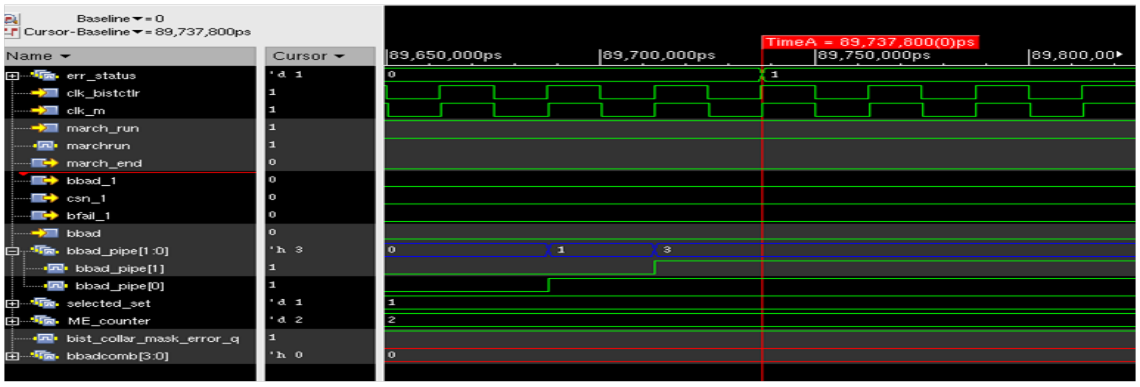
Results

5.1 BUG REPORTS

- **Failure in Diagnosis feature of BIST for SPREG memories**
  - Multiple Runs of the particular tests showed that a failure has occurred in tests for diagnosis.
  - The failing configuration and subsequent march operations are identified using error log files.
  - The results show that the information being shifted out of the BITMAP register containing all information (data, address, fsm states, and algorithm) related with fault occurred is different from the expected register content generated by the checker for collar.
  - Firstly the test bench is checked for self errors if the expected data and address generation is correct.
  - Next for valid reference being generated by test bench ,the RTL for BIST collar is checked for debugging.
  - One of the signals in th RTL is wrongly tied to Logic 1 instead of Logic 0 and this signal is used to mask faults found of previous march operation run and re run the algorithm from beginning.
  - The signal is corrected and correct value (Logic 0) is restored.
  - Faulty configuration is re run and passed successfully.
- **Invalid tests not accounted for in case of sequential sharing of collars of BIST.**
  - False execution status being shifted out by BIST
  - FSM cycles not accounting for invalid tests.
  - Error identified after mismatch in expected status of test bench and actual status shifted out by BIST.

- **Mismatch in implemented Redundancy Scheme w.r.t to memory.**
  - the one implemented in final version of compiler.
  - Order of faulty row replacement by redundant row changed
  - Entire test of redundancy is re written and repair address calculation is changed

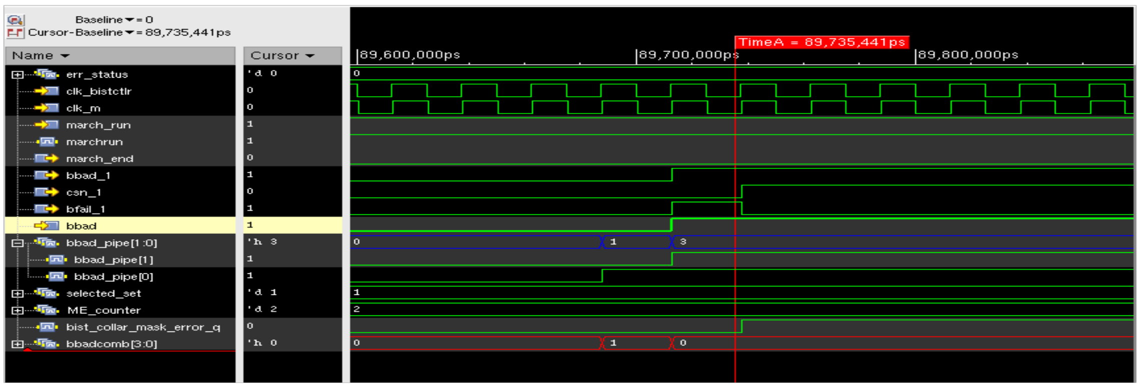
5.1.1 Waveforms modelling some of the errors identified in design.



```
Baseline = 0
Cursor-Baseline = 89,737,800ps
TimeA = 89,737,800(0)ps

Name                       Cursor
err_status                 'd 1
clk_bistctlr               1
clk_m                      1
march_run                  1
marchrun                   1
march_end                  0
bbad_1                     0
csn_1                      0
bfail_1                    0
bbad                       0
bbad_pipe[1:0]             'h 3
bbad_pipe[1]               1
bbad_pipe[0]               1
selected_set               'd 1
ME_counter                 'd 2
bist_collar_mask_error_q   1
bbadcomb[3:0]              'h 0
```

Figure 5.1: Faulty waveform



```
Baseline = 0
Cursor-Baseline = 89,735,441ps
TimeA = 89,735,441ps

Name                       Cursor
err_status                 'd 0
clk_bistctlr               0
clk_m                      0
march_run                  1
marchrun                   1
march_end                  0
bbad_1                     1
csn_1                      0
bfail_1                    1
bbad                       1
bbad_pipe[1:0]             'h 3
bbad_pipe[1]               1
bbad_pipe[0]               1
selected_set               'd 1
ME_counter                 'd 2
bist_collar_mask_error_q   0
bbadcomb[3:0]              'h 0
```

Figure 5.2: Correct waveform

5.2 Improvements in the Test bench

The major accomplishment of the project was to provide a common test bench support for memories with same architecture and technology but different configuration (Speed, Density) and functional features such as redundancy, clock gating etc.
Common support has also been provided for positive and negative edge triggered memories and support for both row redundancy and column redundancy in memories through same compiler for BIST and also tests for same through common test bench.
New tests involving various fault scenarios for repairable memories identified and implemented in the verification test bench.
In the verification environment for BIST Controller, tests for repair data shift and load from collar are modified to accommodate the new architectures of memories and the corresponding redundancy schemes.
The migration of environment from verilog to system verilog is done to cater to the need for advance test and debugging capabilities required in functional verification at IP level.

Conclusions

- The BIST Collar serves as a wrapper to the memory and applies various algorithmic tests provided by BIST Controller to identify various types of faults in memory.
- Verification methodology implemented for BIST Sub Systems is memory specific and test cases are written to verify the various functional and architectural features of BIST.
- Complete verification is accomplished by achieving zero fail run in all three modes of verification i.e. Functional, Gate level functional and Timing simulation and also taking into consideration maximum functional and code coverage.
- The primary aim of applying different methodologies is to verify the design from various levels of abstraction and leave minimum or no scope for any loopholes or mismatch of specification.

Bibliography

[1] Memory BIST user manual @ ST Microelectronics

[2] Memory user manual C28 FDSOI platform

[3] Memory user manual M40 architecture NVM

[4] Zhang, E., Yogev, E. ”Functional Verification using Self Checking Test benches”,Verilog HDL Conference, 1997., IEEE International,March 31 - Apr 3 1997

[5] Cadence Gate level simulation methodology document

[6] Presentation on Advance M-BIST Verification by Samsung Electronics

[7] Language Reference Manual for Verilog HDL

[8] Language Reference Manual for Verilog HDL

[9] Language Reference Manual for System Verilog HVL

[10] Verilog Primer by Samir Palnitkar

[11] System Verilog for Verification (Springer Pub.) Chris Spear

[12] Synopsis Inc. presentation on system verilog and object oriented programming.

[13] www.testbench.in

[14] www.ece.uc.edu

[15] www.asic-world.com/systemverilog