Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 7, July : 2025

FAULT COVERAGE IMPROVEMENT IN MEMORY TESTING: A STUDY ON MARCH (5N) ALGORITHM WITH MBIST

[1]GATTU MANIDEEPAK, [2]Y. RAGHAVENDRA RAO,
[1]M. Tech, VLSI System Design, Student,
Department of ECE,
Email: manideepak015@gmail.com,
[2]Professor, Department of ECE,
Email: yraghavenderrao@gmail.com,
JNTUH University College of Engineering Sultanpur,
Sangareddy, Telangana, India

ABSTRACT
In the rapidly evolving landscape of integrated circuits, ensuring high reliability in memory systems is paramount. This study investigates the effectiveness of the March (5n) testing algorithm when integrated into a Memory Built-In Self-Test (MBIST) framework. The March (5n) algorithm is specifically designed to enhance fault coverage for various memory fault types, including stuck-at and transition faults. The proposed MBIST architecture, which includes a Finite State Machine (FSM) controller, address generator, and data generator, automates the testing process, enabling efficient self-testing of memory devices. Experimental results demonstrate that the integration of the March (5n) algorithm significantly improves fault coverage and reduces testing time compared to conventional testing methodologies. This work underscores the critical role of innovative testing techniques in maintaining the reliability and performance of modern memory technologies.

KEY WORDS: Data Compression, BAQ, Leach Protocol, WSNs.

I. INTRODUCTION

The rapid advancement of digital and embedded systems has intensified the need for dependable semiconductor memory. Memory components—particularly SRAMs and embedded arrays—are integral to the performance and reliability of processors, SoCs, and mobile systems. However, with shrinking technology nodes and increased integration density, memory reliability is increasingly challenged by manufacturing defects, aging effects, and environmental stressors [10]. These factors introduce faults such as stuck-at, transition, and coupling faults, potentially leading to data corruption and system failure. Consequently, comprehensive memory testing is vital during both fabrication and in-field operation.

A. Memory Fault Models

Memory testing begins with fault modelling, which provides a logical abstraction of physical defects. Single-cell faults—like Stuck-At Faults (SAFs), where cells are locked to logic ‘0’ or ‘1’—are typically detected by writing both logic levels and verifying correctness. Transition Faults (TFs), where a cell fails to change state (e.g., $0 \rightarrow 1$), impact timing-sensitive operations and require sequential read/write actions for detection [1], [9]. Data Retention Faults

(DRFs) result from a cell's inability to hold a value over time due to leakage or degradation and are identified using idle periods after write operations [8]. Stuck-Open Faults (SOFs), caused by open circuits, produce undefined outputs and are more difficult to detect using conventional patterns [6].

B. Conventional Testing Techniques and Their Limitations

Traditional memory testing methods rely on Automatic Test Equipment (ATE), which applies externally generated test vectors and monitors outputs. While ATEs provide flexibility and wide fault coverage, they are often costly, slow for large memories, and inadequate for at-speed testing in embedded systems [4]. Additionally, limited pin access and complex routing in SoCs restrict external probing. Software-based diagnostics, executed by embedded processors, offer in-system testing but often lack sufficient fault coverage and real-time applicability. To overcome these limitations, Design-for-Testability (DfT) principles advocate for the integration of test capabilities within the chip. A prominent example is Built-In Self-Test (BIST), which allows on-chip generation and evaluation of test patterns. For memories, Memory BIST (MBIST) specifically targets efficient, autonomous, and at-speed fault detection in embedded arrays [2].

C. March Test Algorithms

March algorithms, widely adopted in MBIST, are structured sequences of memory operations executed in ascending (↑) or descending (↓) address orders. These algorithms—including March C-, March A, and March B—systematically detect SAFs, TFs, and coupling faults with varying coverage and complexity [5]. The March (5n) variant, used in this work, offers a balanced trade-off between fault coverage and hardware simplicity, making it suitable for high-reliability applications.

II. LITERATURE SURVEY

As integrated circuits continue to scale down in size and increase in complexity, embedded memories—particularly SRAMs—have become increasingly vulnerable to faults induced by manufacturing defects, aging, and process variations. Memory testing, therefore, plays a vital role in validating system reliability across all stages of design and deployment. Techniques for testing memories have evolved from traditional external test methods to more efficient built-in approaches that support automation and high fault coverage.

Review on Memory Testing Techniques

A. External Tester-Based Approaches

The conventional method for memory validation involves Automatic Test

Equipment (ATE), which applies controlled test vectors and captures outputs to detect anomalies. ATE systems are versatile and capable of characterizing performance and timing, but they suffer from high cost, long test duration, and limited access to deeply embedded memory blocks in SoCs [10].

B. Built-In Self-Test (BIST)

To address these limitations, Built-In Self-Test (BIST) architectures were developed. BIST incorporates on-chip modules such as test controllers, address generators, data generators, and comparators, enabling the circuit to test itself autonomously. In memory-specific applications, MBIST (Memory BIST) has emerged as a preferred technique due to its low-cost scalability and ability to perform at-speed testing. MBIST significantly reduces dependency on external tools while enhancing test accessibility and time efficiency [4].

C. March Algorithms in Memory Testing

March algorithms represent a class of systematic memory test methods that involve sequences of read/write operations in ascending or descending address orders. These algorithms are designed to expose a broad spectrum of fault types including stuck-at faults (SAFs), transition faults (TFs), and coupling faults (CFs) [10].

- **March C-** (“van de Goor” [11]) is a widely used algorithm performing six operations per memory cell, making it effective for detecting both static and dynamic faults. However, its longer test time limits its applicability in speed-critical systems.
- **March B** (Hamdioui and van de Goor [7]) simplifies the test to five operations per cell, reducing test time while maintaining good SAF and TF coverage, though it is slightly less effective for coupling faults.
- **March Y** (Renovell et al. [3]) is optimized for speed and resource efficiency with only four operations per cell, offering minimal fault coverage suitable for low-power applications.
- **March SS** (Hamdioui [12]) extends fault detection to complex state-coupling faults using a six-operation scheme. Its broader coverage makes it ideal for safety-critical systems, albeit with increased hardware overhead.
- **March LA** (Wang and Gupta [5]) departs from linear address progression, targeting address decoder faults through dynamic linking of test addresses. While highly effective for decoder-related faults, its complexity poses challenges in MBIST design.
- **March (5n)** (Paschalis and Gizopoulos [6]) offers a well-balanced solution with five operations per cell, delivering strong detection capability across SAFs, TFs, and CFs with reduced complexity. Its deterministic flow and efficiency make it

suitable for FSM-based MBIST architectures in embedded environments.

III. PROPOSED METHODOLOGY

As embedded memories dominate the silicon area of modern integrated circuits, ensuring their reliability is essential for system correctness. Traditional memory test techniques, especially those relying on external Automatic Test Equipment (ATE), are often cost-prohibitive, time-intensive, and lack the granularity needed for testing deeply embedded memory components. To address these limitations, this work employs a Memory Built-In Self-Test (MBIST) framework driven by the March (5n) algorithm—known for its efficiency and high fault coverage. The proposed methodology encompasses memory fault modelling, modular MBIST architecture development, algorithm implementation using Verilog HDL, and validation through simulation and fault injection analysis.

A. Memory Fault Modelling

Effective memory testing begins with accurate modelling of potential fault types. Key fault classes considered include:

- **Stuck-at Faults (SAFs):** Permanent logic errors where a cell is fixed at '0' or '1'.
- **Transition Faults (TFs):** Failure in switching states (0→1 or 1→0).
- **Coupling Faults (CFs):** Logical interference between neighbouring cells.
- **Address Decoder Faults (ADFs):** Errors resulting from faulty or aliased address lines.

These fault models serve as a benchmark for evaluating the coverage capabilities of the March (5n) test. Fault conditions are introduced in the memory model to validate the algorithm's effectiveness during simulation.

B. MBIST Architecture Design Using March (5n)

The MBIST system integrates a structured set of hardware components that coordinate test operations autonomously, without external control as shown in Figure 1. The architecture is built around six key modules:

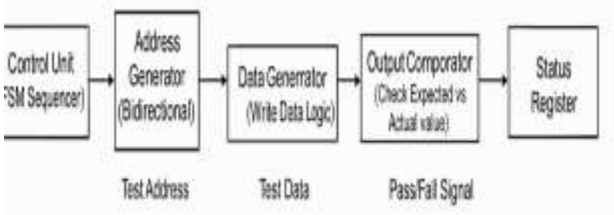
Control Unit
FSM Sequencer)
Address Generator (Bidirectional)
Data Generator (Write Data Logic)
Output Comporator (Check Expected vs Actual value)
Status Register
Test Address
Test Data
Pass/Fail Signal

Figure 1. Architecture design of March (5n)

- **Finite State Machine (FSM) Controller:** The Finite State Machine (FSM) functions as the control core of the MBIST system, directing the execution of the March (5n) algorithm through a well-defined sequence of states. Figure 2 represents the state diagram of FSM controller. It manages operations such as writing ‘0’, reading

‘0’, writing ‘1’, and their execution in both ascending and descending memory address orders. The FSM begins in the IDLE state, awaiting the activation signal. Once triggered, it transitions through states including W0_ASC (write ‘0’ in ascending order), R0W1_ASC (read ‘0’ and write ‘1’ in ascending order), R1W0_DESC (read ‘1’ and write ‘0’ in descending order), R0_ASC (read ‘0’ in ascending order), and R0_DESC (read ‘0’ in descending order), concluding with the DONE state, which signals test completion. Each state activates specific control signals for modules such as the address generator, data generator, and comparator to perform the required operation.

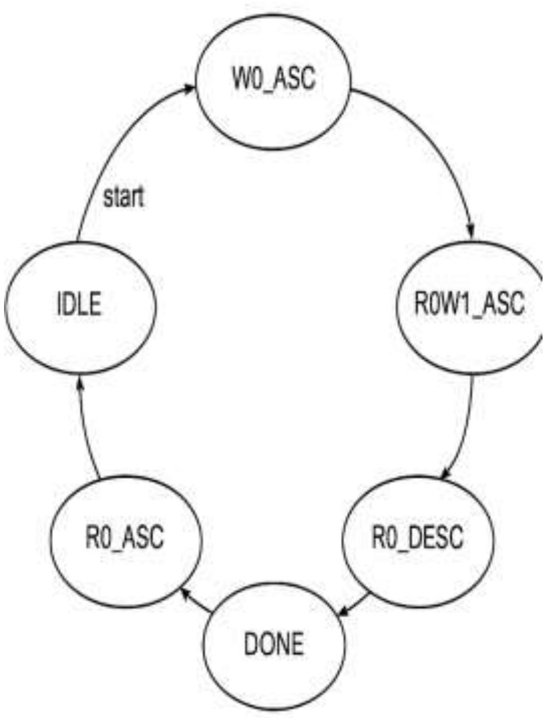

Figure 2. State diagram of FSM controller.

The FSM logic is implemented using a case construct in Verilog HDL, and transitions are synchronized with the system clock. Additionally, optional clock gating is employed to disable inactive modules during non-operational states, enhancing power efficiency.

- **Address Generator:** A bidirectional counter generates memory addresses in ascending and descending order, as required by the March (5n) test phases. The counter is parameterizable for different memory sizes and ensures full address coverage through direction control and wrap-around logic.
- **Data Generator:** This block supplies static logic values ('0' and '1') during write operations. Controlled by FSM signals, it ensures correct data is applied for each test phase. While basic in design, the generator can be extended to support pseudorandom patterns for enhanced fault stimulation.
- **Comparator Module:** The comparator monitors memory read outputs and checks for mismatches against expected values. If a discrepancy is detected, a fault flag is raised. Operating synchronously with the system clock, the comparator is essential for identifying both permanent and transient memory faults during read cycles.

IV. IMPLEMENTATION OF MARCH (5N)

The March (5n) algorithm is a lightweight yet effective memory testing technique widely adopted in MBIST frameworks due to its balance between fault coverage and test complexity. It performs five specific memory operations across all addresses in both ascending and descending orders, enabling detection of stuck-at, transition, and coupling faults with reduced hardware overhead.

A. March (5n) Test Elements:

- ↑(w0): Write logic '0' to all addresses in ascending order.
- ↑ (r0, w1): Read '0', then write '1' in ascending order.
- ↓ (r1, w0): Read '1', then write '0' in descending order.
- ↑ (r0): Read '0' in ascending order.
- ↓ (r0): Read '0' in descending order.

B. Step-wise Working:

- **Initialization:**
  - MBIST enters IDLE state.
  - Resets memory, FSM, address counter, and fault flags.
  - Test begins when start_bist = 1.
- **Step 1 – ↑(w0):**
  - FSM moves to W0_ASC.
  - All memory locations are sequentially written with '0'.
  - Example: MEM [0] = 0, MEM [1] = 0, ..., MEM [N] = 0.
- **Step 2 – ↑ (r0, w1):**
  - FSM transitions to R0W1_ASC.
  - Each address is read and compared with expected '0', then overwritten with '1'.
  - Fault is flagged if mismatch occurs.
- **Step 3 – ↓ (r1, w0):**
  - FSM enters R1W0_DESC.
  - Reverse traversal; read '1' and write back '0'.
  - Confirms previous write success and checks for transition faults.
- **Step 4 – ↑ (r0):**
  - FSM moves to R0_ASC.
  - Ascending pass to read and verify all cells are reset to '0'.
- **Step 5 – ↓ (r0):**
  - FSM enters R0_DESC.
  - Final descending read pass ensures fault coverage in reverse order.
- **Completion:**
  - FSM moves to DONE.
  - done = 1 and fail = 1 if faults detected, with address stored.
  - Ensures the test cycle is complete and results are ready for logging or debugging.

This structured approach highlights each operational phase of the March (5n) algorithm while emphasizing the role of FSM in sequencing, control signal activation, and fault flagging. It suits journal formatting where clarity, technical correctness, and brevity are essential.

V. RESULTS:

The simulation waveform illustrated in Figure 3, 4, 5, presents a successful execution of the March (5n) test algorithm on a memory block using the proposed MBIST architecture. The timing diagram confirms that each module—FSM controller, address generator, data generator, and comparator—functions in synchronization, achieving the test goals.

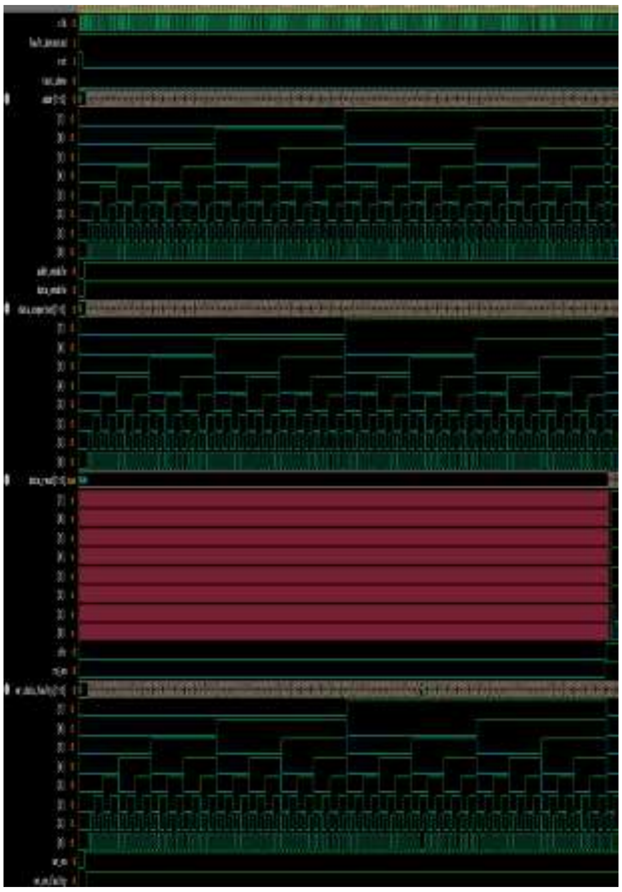
Figure 3. Simulation Result-1

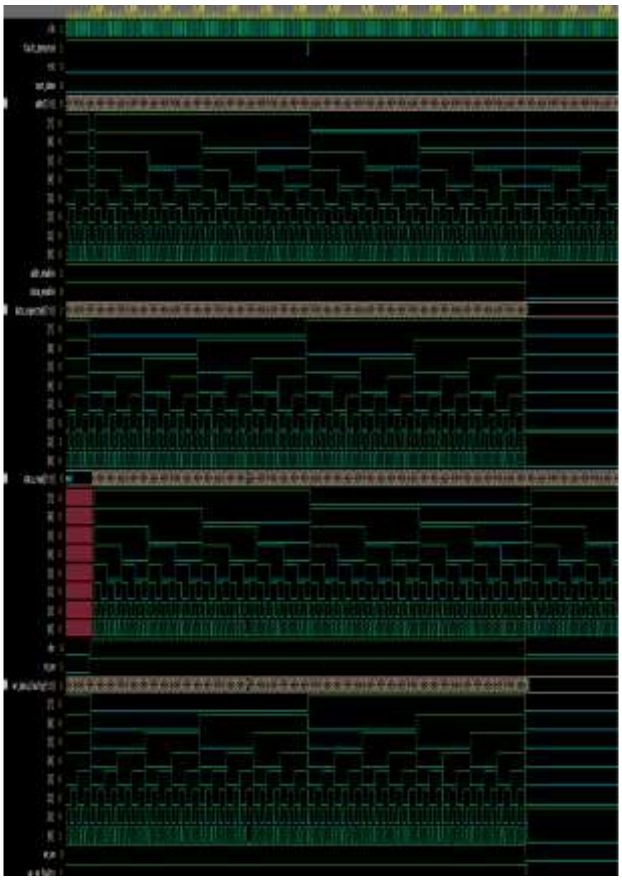
Figure. 4 Simulation Results-2

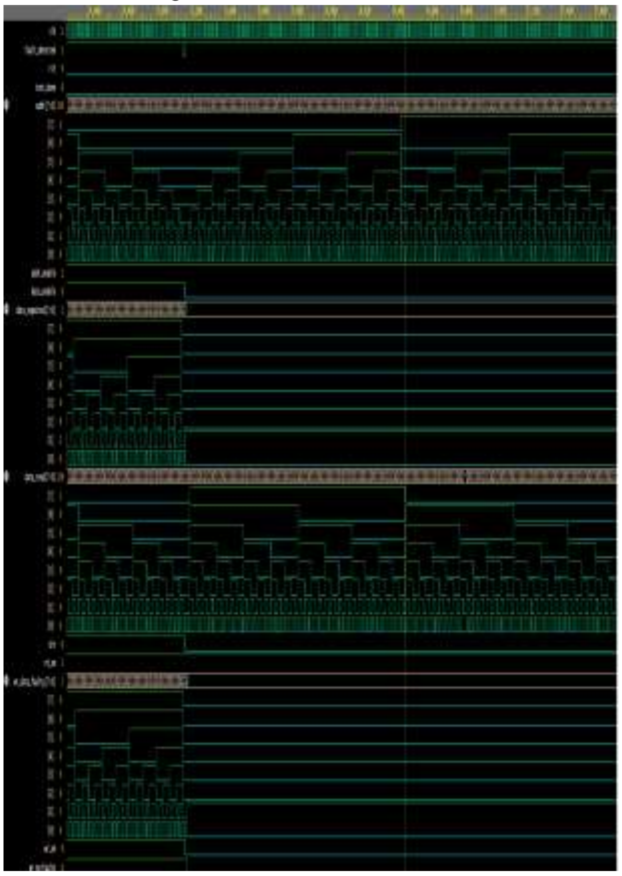
Figure. 5 Simulation Result-3

VI. CONCLUSION

In conclusion, the integration of the March (5n) algorithm within a modular MBIST framework offers a robust and cost-effective solution for embedded memory testing in modern SoCs. The architecture—comprising an FSM controller, bidirectional address generator, data generator, and comparator—executes structured test sequences that effectively identify stuck-at and transition faults. Simulation results validate functional correctness and demonstrate reliable fault detection through synchronous read/write operations and status signalling. The design minimizes hardware overhead while maximizing test coverage, making it scalable and reusable for various memory configurations without reliance on external testers.

VII. FUTURE SCOPE

Despite its proven efficiency, the proposed MBIST design offers scope for further enhancement:

- **Expanded Fault Detection:** Future iterations can include support for complex fault types like bridging, retention, and coupling faults through algorithm extensions or hybrid techniques.
- **Built-In Repair (BIRA):** Incorporating redundancy and self-repair logic can enhance memory reliability and yield.
- **Low-Power Optimization:** Techniques like clock gating can reduce power consumption, beneficial for battery-operated and IoT applications.
- **Hardware Validation:** Implementation on FPGAs or ASICs with full DFT support would enable real-world deployment.
- **Multi-Port Memory Support:** Adapting the design for multi-bank or dual-port memories addresses the needs of high-performance applications.
- **Standards Integration:** Compliance with IEEE 1149.1 (JTAG) or IEEE 1500 can improve interoperability with industry test infrastructures.

REFERENCES

JOURNALS:

[1]. B. Ben Romdhane, H. Amar, and M. Renovell, "SRAM Memory Test Techniques: A Survey," Microelectronics Journal, vol. 45, no. 2, pp. 193–203, 2014.

[2]. M. Nicolaidis, "Design for Soft Error Mitigation," IEEE Transactions on Device and Materials Reliability, vol. 5, no. 3, pp. 405–418, 2005.

[3]. M. Renovell, Y. Zorian, and J. Figueras, "Modeling and Testing of Memory Faults," IEEE Design & Test of Computers, vol. 18, no. 3, pp. 56–64, May–June 2001.

[4]. S. Wang and S. K. Gupta, "DS-LFSR: A New BIST TPG for Low Heat Dissipation," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 21, no. 7, pp. 842–851, 2002.

[5]. S. Wang and S. K. Gupta, “ATPG for Embedded SRAMs,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 22, no. 7, pp. 885–898, 2003.

[6]. A. Paschalis and D. Gizopoulos, "Effective Built-In Self-Test for Word-Oriented RAMs," IEEE Design & Test of Computers, vol. 19, no. 3, pp. 74–83, May–June 2002.

CONFERENCES:

[7]. A. J. van de Goor, “March Tests for Realistic Static Faults in RAMs,” Proc.

IEEE Int’l Test Conference, pp. 272–281, 2000.

[8]. P. Girard, C. Landrault, S. Pravossoudovitch, and D. Duval, "A Test for Data Retention Faults in SRAMs," Proc. IEEE Int'l Workshop on Memory Technology, Design and Testing (MTDT), pp. 93–98, 1996.

[9]. S. Hamdioui, A. J. van de Goor, and K. K. Saluja, "A Novel Transition Fault Test for Embedded SRAMs," IEEE Asian Test Symposium, pp. 182–187, 2000.

[10]. Y. Zorian, "A Distributed BIST Control Scheme for Complex VLSI Devices," Proc. IEEE VLSI Test Symposium, pp. 4–9, 1993.

BOOKS:

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

[12]. S. Hamdioui, RAM Test Algorithms for Embedded Systems, Ph.D. dissertation, Delft University of Technology, 2001.