Modified March FSM-Based Memory BIST Architecture

Ahmed Salahuddin Suhaib[1]; Dr. M. Asha Rani[2]

[1,2]Electronics and Communication Engineering Jawaharlal Nehru Technological University Hyderabad Hyderabad, India.

Publication Date: 2025/09/04

Abstract: The Memory Built-In Self-Test (MBIST) is the standard for testing dense embedded memories that dominate modern SoCs; however, a critical trade-off exists between the test time and fault coverage. While comprehensive algorithms such as March C- (10n) are slow, faster algorithms such as MATS++ (6n) are often preferred, although both aim to detect critical Address Decoder Faults (AFs). This study presents an MBIST controller employing a novel March (5n) algorithm that bridges this gap, offering robust fault coverage with superior efficiency. The core innovation of the algorithm is the "address-as-data" paradigm, which uses the memory address (a) and its bitwise complement (~a) as test patterns to efficiently detect Stuck-at (SAF), Transition (TF), and Address Decoder (AF) faults.

The proposed FSM-based controller has been designed in Verilog and validated on a Xilinx Zynq-7000 series FPGA platform. Experimental evaluation demonstrates that the March (5n) algorithm achieves significant reductions in test time compared to established approaches, with minimal resource overhead. These findings highlight the effectiveness of the March (5n) algorithm in achieving a balanced trade-off between speed and fault coverage, positioning it as a practical candidate for deployment in high-volume, cost-sensitive applications.

Keywords: *MBIST, March 5n, March C-, MATS++.*

How to Cite: Ahmed Salahuddin Suhaib; Dr. M. Asha Rani (2025) Modified March FSM-Based Memory BIST Architecture. *International Journal of Innovative Science and Research Technology,* 10 (8), 2165-2169. https://doi.org/10.38124/ijisrt/25aug1309

I. INTRODUCTION

The scaling of system-on-chip (SoC) technology has made embedded memories the dominant on-chip component, creating significant testing challenges due to their density and limited accessibility. Traditional external testing with Automatic Test Equipment (ATE) is costly and often impractical, making Memory Built-In Self-Test (MBIST) the industry-standard solution. By integrating test logic on-chip, MBIST enables efficient, at-speed testing while reducing dependence on external hardware.

The effectiveness of MBIST depends largely on the underlying algorithm, where a trade-off exists between fault coverage and test time. March-based algorithms are widely adopted but illustrate this conflict: comprehensive tests such as March C- (10n) achieve high fault coverage, including Stuck-at (SAF), Transition (TF), Address Decoder (AF), and Coupling Faults (CF), but incur long execution times. Faster algorithms such as MATS++ (6n) reduce test time and still cover SAF, TF, and AFs, but fail to detect the broader range of faults. This motivates the need for an intermediate solution that achieves critical fault coverage with lower complexity.

This work introduces an FSM-based MBIST controller implementing a novel March (5n) algorithm that applies an "address-as-data" paradigm, using each memory address and its bitwise complement as test patterns. This design achieves robust AF coverage comparable to MATS++ but within a 5n complexity framework, improving efficiency without significant overhead. The controller is validated through fault injection simulations and FPGA hardware implementation on a Xilinx Zynq-7000 device, with results benchmarked against March C- and MATS++. The findings demonstrate that the March (5n) algorithm offers a balanced solution between speed and coverage, making it well-suited for high-volume, cost-sensitive applications.

II. BACKGROUND AND RELATED WORK

To fully appreciate the contribution of the March (5n) algorithm, it is essential to first understand the types of physical defects it targets and the established algorithms against which it is benchmarked. This section provides an overview of common memory fault models and summarizes the principles of industry-standard March test algorithms.

➢ *Memory Fault Models*

The dense, array-based structure of modern memories makes them susceptible to a unique set of physical defects that differ from those in random logic. These defects are abstracted into functional fault models, which describe the incorrect behaviour of the memory from an external perspective. A test algorithm's quality is measured by its ability to detect these modelled faults. The key fault models relevant to this study are:

- Stuck-at Faults (SAF): A cell or line is fixed at 0 or 1 regardless of input. Detection requires writing and reading both values.
- Transition Faults (TF): A cell fails to switch from 0→1 or 1→0, typically due to timing issues. Tests must enforce both transitions.
- Address Decoder Faults (AF): Faulty decoding may cause a cell to be unaddressable, multiple cells to be accessed, or the same cell to appear at different addresses. These faults require sequences in opposite address orders.
- Coupling Faults (CF): An operation on one cell (aggressor) alters another cell (victim), such as inversion or forcing to a constant state. Detecting CFs requires more complex data backgrounds.

➢ *March Test Algorithms*

March tests are the dominant methodology for memory testing due to their linear complexity and high fault coverage. A March test consists of a sequence of "March elements," where each element specifies a series of read/write operations performed on every cell, typically in an ascending (↑) or descending (↓) address order, before moving to the next element. The following algorithms are the benchmarks for this study:

- MATS++ (6n): The MATS++ algorithm is represented as {↑(w0); ↑ (r0, w1); ↓ (r1, w0); ↑(r0)}. It efficiently detects SAF, TF, and AF with six operations per cell.
- March C- (10n): Represented as {(w0); ↑ (r0, w1); ↑ (r1, w0); ↓ (r0, w1); ↓ (r1, w0); ↑(r0)}, March C- is one of the most widely used algorithms covering SAF, TF, AF, and many CFs. Its thoroughness comes at the cost of long test times.

These algorithms illustrate the core MBIST trade-off: March C- offers broad coverage but is slow, while MATS++ is faster but limited. This gap motivates intermediate approaches like the March (5n) algorithm, which aims to balance efficiency with robust AF detection.

III. THE MARCH (5N) ALGORITHM AND MBIST ARCHITECTURE

The proposed MBIST solution introduces a Finite State Machine (FSM)-based controller implementing a novel March (5n) algorithm. The aim is to achieve reliable fault coverage with reduced test time compared to conventional March tests. This section outlines the algorithm’s operational principles, the controller architecture, and the FPGA-based implementation used for validation.

➢ *Algorithmic Principles*

The March (5n) algorithm consists of four elements: {↑(wb, wa); ↑(ra); ↓(wb); ↓(rb)}, where a represents the memory address, and b its bitwise complement. Unlike traditional tests that rely on static 0/1 patterns, this “address-as-data” paradigm generates dynamic data backgrounds directly from the address space.

- *This Structure Enables Efficient Fault Detection:*

✓ SAF: By writing and reading both a and its bitwise complement b for every memory location, every bit in the data word is subjected to both a 0 and a 1 state, ensuring any stuck-at condition is exposed.
✓ TF: The (wb,wa) element sensitizes the up-transition, which is verified by the subsequent ra operation. The transition from a to b (between the ra and wb elements) sensitizes the down-transition, which is verified by the final rb operation.
✓ AF: A fault where a higher address incorrectly accesses a lower one is caught by the initial ascending pass (↑(ra)). Conversely, a fault where a lower address overwrites a higher one is masked during the ascending pass but is guaranteed to be detected by the subsequent descending pass (↓(rb)). This intelligent structure ensures comprehensive AF coverage.

Thus, the algorithm achieves full SAF, TF, and AF coverage within 5n complexity.

➢ *Hardware Architecture*

The controller is implemented as a modular FSM-based design with four main blocks:

- FSM Controller: Sequences the March elements (M0–M3) and synchronizes operations.
- Address Generator: Produces ascending or descending addresses using a counter and signals pass completion.
- Data Generator: Derives a and b patterns. For wider data buses, an address-tiling scheme replicates address bits across the word, ensuring all bits are exercised.
- Memory Interface: Switches the MUT into test mode by routing control, address, and data signals from the MBIST controller.

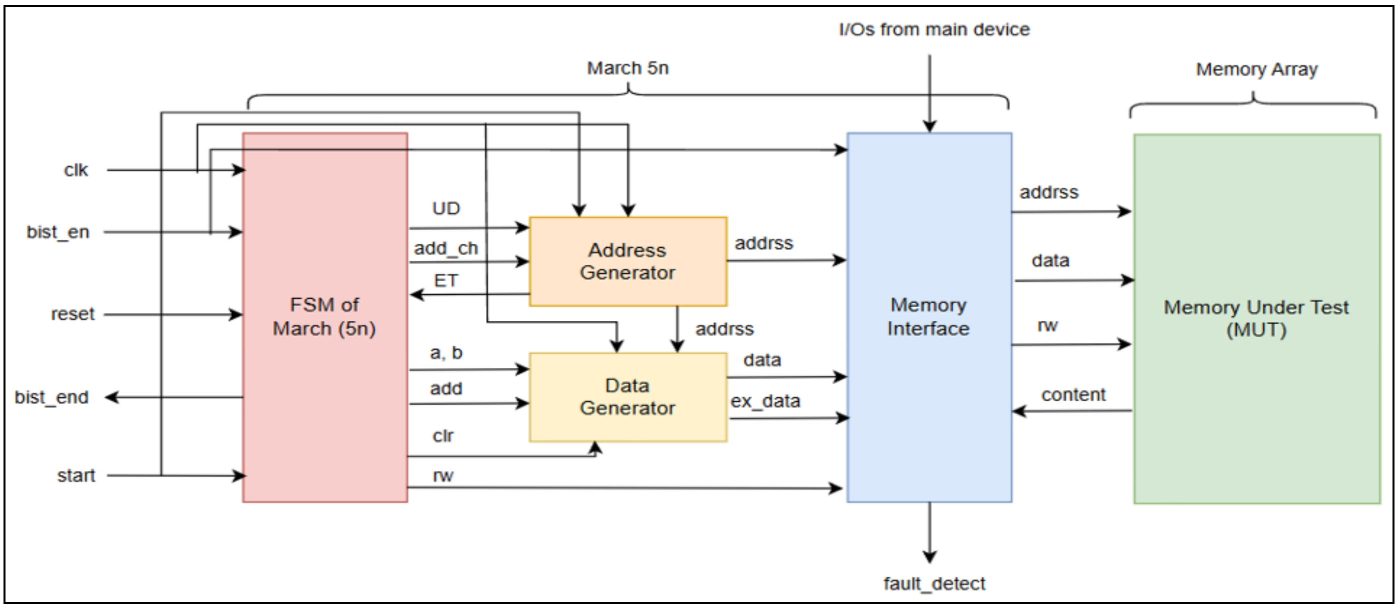

Fig. 1 High Level MBIST Controller Design

➢ *Implementation Details*

The design was coded in Verilog and deployed on a Xilinx Zynq-7000 FPGA using Vivado. A Zedboard interface with switches and LEDs supports reset, test initiation, fault injection, and real-time status reporting. This platform demonstrates correct functionality and validates the proposed architecture under hardware conditions.

IV. EXPERIMENTAL SETUP AND METHODOLOGY

The March (5n) MBIST controller was validated through simulation-based fault coverage analysis and FPGA-based hardware evaluation. The methodology included controlled fault injection in simulation and comparative benchmarking against March C- and MATS++ to assess coverage, performance, and implementation cost.

➢ *Test Environment*

Design and functional verification were carried out in Verilog using a standard HDL simulator. Fault coverage was analyzed at the simulation level, while hardware synthesis and implementation were performed with the Xilinx Vivado Design Suite. A Xilinx Zynq-7000 FPGA on the Digilent Zedboard served as the validation platform, enabling collection of real hardware metrics including area, power, and execution time.

➢ *Fault Injection and Coverage Analysis*

Fault detection capability was evaluated using a fault injection framework. A total of 100 faults were modeled: 20 Stuck-at, 20 Transition, 30 Address Decoder (10 each for non-access, multiple-access, and wrong-access), and 30 Coupling (10 each for dynamic, inversion, and idempotent). Each fault was injected individually into the memory model and tested with the controller. Coverage was computed as the percentage of injected faults successfully detected by the algorithm, providing a uniform basis for comparison.

➢ *Performance and Resource Evaluation*

Beyond coverage, the controllers were benchmarked on three implementation metrics for memories of 4KB, 8KB, 32KB, and 64KB:

- Resource Utilization: FPGA area cost, measured in LUTs, Flip-Flops, and BRAMs.
- Power Consumption: On-chip dynamic power reported by Vivado.
- Test Time: Clock cycles and execution time required to complete the test sequence, giving an implementation-independent measure of efficiency.

V. RESULTS AND ANALYSIS

The performance of March C-, MATS++, and the proposed March (5n) MBIST controllers was evaluated through simulation and FPGA implementation. Results are presented in terms of fault coverage and hardware metrics, with emphasis on comparative efficiency.

➢ *Fault Coverage Analysis*

Table 1 Summarizes the Coverage Achieved Against 100 Injected Faults.

Table 1 Fault Coverage Comparision

Algorithm	Complexity	Faults Detected	Coverage
March C-	10n	SAF, TF, AF, CF	100%
MATS++	6n	SAF, TF, AF	70%
March (5n)	5n	SAF, TF, AF	70%

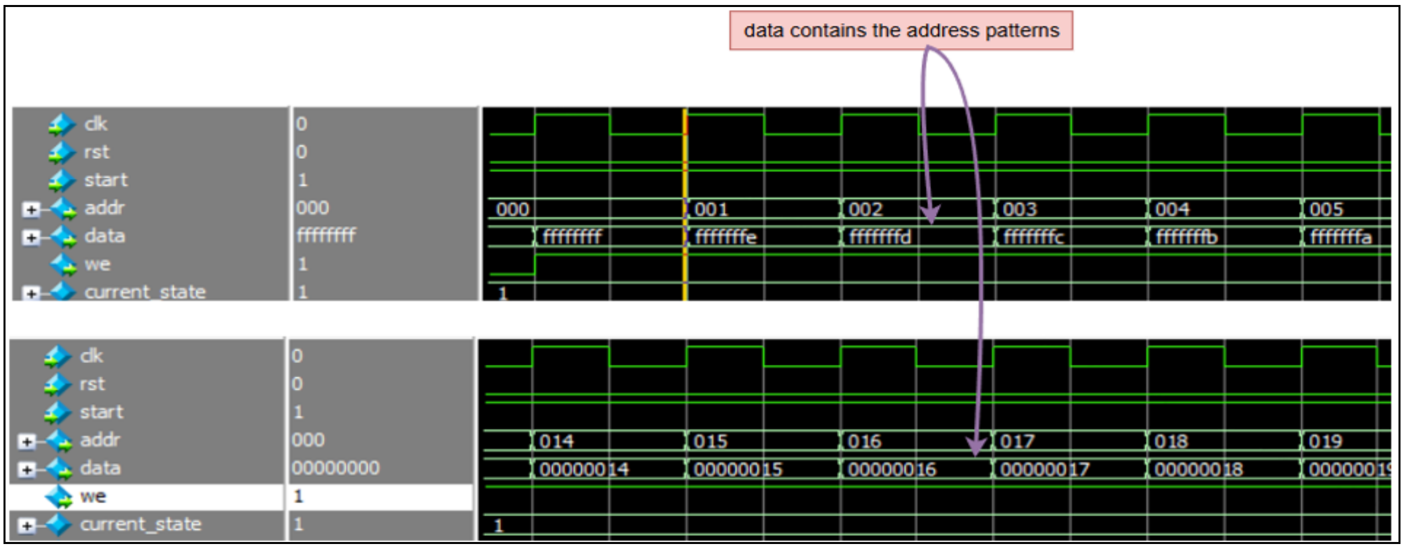

Fig 2 ModelSim Simulation of March 5n

March C- confirmed its status as the most comprehensive test, detecting all fault classes. Both MATS++ and March (5n) achieved 70% coverage, targeting SAF, TF, and AF but not Coupling Faults. The equivalence in fault coverage between March (5n) and MATS++ highlights that the key differentiator lies in test efficiency rather than detection capability.

➢ *Performance and Resource Analysis*

Hardware synthesis results for the three controllers across multiple memory sizes are given in Tables 2–4.

Table 2 March C- Performance and Resource Utilization

Metric	4KB	8KB	32KB	64KB
LUTs	31	28	30	31
FF	55	56	58	59
Power (W)	0.121	0.118	0.118	0.117
Clock Cycles	10,245	20,485	81,925	163,845

Table 3 MATS++ Performance and Resource Utilization

Metric	4KB	8KB	32KB	64KB
LUTs	30	28	29	30
FF	52	53	55	56
Power (W)	0.115	0.112	0.113	0.115
Clock Cycles	6,146	12,290	49,154	98,306

Table 4 March (5n) Performance and Resource Utilization

Metric	4KB	8KB	32KB	64KB
LUTs	30	28	29	30
FF	50	51	53	53
Power (W)	0.113	0.113	0.114	0.114
Clock Cycles	5,124	10,244	40,964	81,924

- Test Time: The March (5n) controller consistently outperformed the other algorithms. Compared to March C-, it reduced test time by about 50% (e.g., 10,245 vs. 5,124 cycles at 4KB). Against MATS++, it delivered an average 20% speed improvement while maintaining identical fault coverage.
- Resource Utilization: LUT and FF usage across all algorithms was nearly identical, with differences of only a few registers.
- Power Consumption: Power measurements remained within a narrow range (0.112–0.121 W), showing no meaningful penalty from the address-as-data mechanism.

VI. DISCUSSION

The results demonstrate that March (5n) achieves the same coverage as MATS++ while reducing test time by 20%. At the same time, it maintains the lightweight hardware profile typical of linear March tests. Compared with March C-, it offers a practical compromise—delivering critical SAF, TF, and AF detection at half the time and negligible resource savings. For cost-sensitive applications where AF coverage is essential but full 10n testing is impractical, the March (5n) controller provides a balanced and efficient solution.

VII. CONCLUSION

This work addressed the trade-off between test time and fault coverage in Memory Built-In Self-Test by introducing an FSM-based controller for the March (5n) algorithm. The design was implemented in Verilog, verified through fault injection, and validated on a Xilinx Zynq-7000 FPGA. Comparative evaluation against March C- and MATS++ confirmed the effectiveness of the approach.

The results show that March (5n) achieves 70% coverage of common fault models—equivalent to MATS++—while reducing test time by approximately 20%. Against March C-, it maintains critical SAF, TF, and AF detection at half the test time. These gains are achieved without additional resource or power overhead, demonstrating that the address-as-data principle is both practical and efficient.

Overall, the March (5n) algorithm offers a balanced alternative for applications where Address Decoder Fault coverage is essential but the time cost of a full 10n algorithm is unacceptable. Its efficiency and low hardware footprint make it well-suited for cost-sensitive, high-volume memory testing.

REFERENCES

[1]. T. S. Nguan Kong, N. E. Alias, M. L. P. Tan, A. Hamzah, U. U. Sheikh, I. Kamisian, and Y. A. Wahab, "An Efficient March (5n) FSM-Based Memory Built-In Self-Test (MBIST) Architecture," in Proc. 2021 IEEE Regional Symposium on Micro and Nanoelectronics (RSM), 2021, pp. 76-79.

[2]. B. Singh, A. Khosla, and S. B. Narang, "Area Overhead and Power Analysis of March Algorithms for Memory BIST," *Procedia Engineering*, vol. 30, pp. 930-936, 2012.

[3]. M. Parvathi, N. Vasantha, and K. S. Prasad, "Modified March C - Algorithm for Embedded Memory Testing," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 2, no. 5, pp. 571-576, Oct. 2012.

[4]. N. Q. M. Noor, Y. Yusof, and A. Saparon, "Low Area FSM-Based Memory BIST for Synchronous SRAM," in *Proc. 5th International Colloquium on Signal Processing & Its Applications (CSPA)*, 2009, pp. 409-412.

[5]. J. Kruthika, G. R. Nisha, R. Gayathri, and V. Jeyalakshmi, "SRAM Memory Built in Self-Test using MARCH Algorithm," in *Proc. 2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, 2022, pp. 1288-1292.

[6]. L. H. R, Varchaswini R., and Y. J. M. Shirur, "Implementation of FSM-MBIST and Design of Hybrid MBIST for Memory cluster in Asynchronous SoC," *International Journal of Computer Applications Technology and Research*, vol. 3, no. 4, pp. 216-220, 2014.

[7]. M. Mamatha and M. Muralidhar, "Memory Testing using March C-Algorithm," *International Journal of VLSI System Design and Communication Systems*, vol. 2, no. 7, pp. 512-517, Oct. 2014.

[8]. D. Jariwala and P. Garg, "FSM Based Memory BIST using Verilog-HDL," Dept. of Electronics and Communications Engineering, Nirma University, Ahmedabad, India, 2022.