International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
© 2026 IJERST. All Rights Reserved
Peer Reviewed, Referred & Indexed Journal

Research Paper

Design of Hybrid Memory Logic Based BIST for Memory Testing

M. Chandrakala
PG Scholar, Department of Electronics and Communication Engineering
National Institute of Technical Teachers Training and Research
Chennai, India
chandrakalamandhodi@gmail.com

Dr. P. Sivasankar
Professor, Department of Electronics and Communication Engineering
National Institute of Technical Teachers Training and Research
Chennai, India
sivasankar@nitttrc.ac.in

Prof. Dr. G. Kulanthaivel
Professor and Head, Department of Electronics and Communication Engineering
National Institute of Technical Teachers Training and Research
Chennai, India
gkvel@nitttrc.ac.in

Abstract— Embedded memories occupy a dominant fraction of contemporary system-on-chip area and are increasingly exposed to manufacturing variation, timing-sensitive defects, coupling effects, and in-field degradation. This paper presents a Hybrid Memory Logic Based Built-In Self-Test (HML-BIST) architecture that combines deterministic March operations with logic-BIST-style linear-feedback shift-register patterns in a unified finite-state-machine-controlled framework. The design contains a dual-mode test-pattern generator, ascending/descending/random address generation, a 256 × 8 synchronous SRAM under test, a bitwise response analyzer, and a location-aware fault register. Deterministic passes target stuck-at, transition, coupling, address-decoder, retention, and read/write faults, while pseudorandom stress targets neighborhood pattern-sensitive and dynamic faults. The Verilog RTL was evaluated using Xilinx Vivado 2023.1 XSim and synthesized for an Artix-7 XC7A100T FPGA. The implementation uses 187 LUTs, 94 flip-flops, one RAMB18E1 block, and no DSP resources, closes timing at 200 MHz, and reports 42.0 ms test time, 12.4 mW dynamic power, and 3.2% peripheral-logic area overhead. Recalculation of the detailed injection table gives 3,948 detections from 4,096 injected instances, corresponding to 96.4% count-consistent overall coverage across eight fault categories. The architecture therefore provides a compact, at-speed, fault-localizing platform suitable for reusable embedded-memory test and future safety-oriented SoC integration.

Keywords— Built-in self-test, Memory BIST, Logic BIST, March algorithm, LFSR, SRAM testing, Fault coverage, Verilog HDL, Artix-7 FPGA.

I. INTRODUCTION

Embedded SRAMs, register files, caches, and local buffers now account for a large fraction of the silicon area in processors and system-on-chip devices. As feature sizes shrink, the distance between storage nodes, word lines, and bit lines becomes small enough that local process variation, resistive opens, shorts, leakage, and capacitive interference can alter cell behavior. A memory that passes a low-speed external test may still fail when exercised at its operational clock, after a retention interval, or under a particular neighboring-data configuration. These characteristics make embedded-memory testing a central design-for-testability problem rather than a final manufacturing step [1]–[3].

External automatic test equipment provides accurate stimulus and measurement, but access to deeply embedded arrays is limited by package pins, test bandwidth, and tester frequency. On-chip BIST places the controller, pattern source, address sequencer, and response checker next to the memory, allowing at-speed production test, power-on self-test, and scheduled in-field health checks. The design problem is to obtain high fault coverage without excessive test cycles, switching activity, or area overhead. Deterministic March algorithms guarantee detection of defined static fault classes, whereas pseudorandom sequences explore a larger state space with compact hardware but provide probabilistic coverage and can hide fault location when only signature compaction is used [3], [4].The proposed architecture combines these complementary strategies. A deterministic phase writes and reads structured backgrounds in ascending and descending order. A pseudorandom phase uses an 8-bit maximal-length Galois LFSR to generate 255 nonzero data states and a second uncorrelated sequence for noncontiguous addresses. The response analyzer compares every read value with its expected value, and the fault register stores the first failing address, error vector, classified fault type, and total number of failures. This preserves diagnostic information required by built-in redundancy analysis and physical failure analysis.

The principal contributions are: (1) a parameterizable hybrid controller that coordinates March and pseudorandom tests in one session; (2) coverage of eight project-defined fault categories through complementary address, data, timing, and retention stress; (3) synthesisable Verilog RTL verified by directed fault injection; (4) resource, timing, power, and test-time analysis on an Artix-7 FPGA; and (5) correction of arithmetic inconsistencies in the phase report so the conference results remain reproducible from the detailed tables.

II. RELATED WORK

March testing was formalized as an ordered sequence of memory operations applied to every address [1]. The March C family provides linear test complexity and strong detection of stuck-at, transition, and coupling faults [2]. Subsequent work expanded the functional fault model to include decoder faults, read/write disturbances, retention behavior, linked faults, and dynamic faults that require more than one operation to sensitize [5]–[10]. March tests are attractive because their detection properties can be reasoned about directly, but a fixed sequence may miss neighborhood combinations or require several independent passes to cover all target classes.

LFSR-based BIST generates long pseudorandom sequences with a small XOR network and a register. It is widely used in logic BIST and can supplement memory tests by varying both data backgrounds and access order [3], [11], [12]. The main concerns are random-pattern-resistant faults, signature aliasing, and the absence of deterministic diagnosis.

Hybrid approaches therefore combine a March kernel with pseudorandom or programmable phases [4], [13], [14]. Power-aware BIST additionally controls switching activity because simultaneous address, data, and memory transitions can exceed the normal functional power envelope [15].

Safety-oriented applications require evidence that diagnostic mechanisms cover the relevant failure modes and operate within a defined fault-tolerant time interval. ISO 26262 motivates high diagnostic coverage for automotive memories, while IEEE 1500 and IEEE 1149.1 standardize access and test infrastructure for embedded cores and boards [16]–[18]. FPGA prototyping is valuable before ASIC implementation because it validates the state machine, pattern sequencing, fault logging, and timing behavior using a repeatable hardware platform [19]–[22].

Table 1. Comparison of Baseline Memory-Test Approaches

Approach	Coverage	Test time	Principal limitation
External ATE	~55%	~120 s	Tester speed and limited embedded access
MATS+ BIST	81.2%	31.5 ms	Targets mainly SAF and TF
March C- BIST	89.3%	65.5 ms	No full NPSF/dynamic coverage
LFSR BIST	85.6%	29.0 ms	Probabilistic; weak diagnosis
Programmable BIST	93.8%	57.0 ms	Higher area and repeated passes

III. MATERIALS AND METHODS

A) System Organization

The Hybrid BIST boundary is inserted between the normal memory interface and the memory under test. When test_enable is low, address, write data, read enable, and write enable originate from functional logic. When test_enable is asserted, multiplexers transfer control to the BIST controller. The architecture in Fig. 1 contains the FSM control unit, deterministic data generator, LFSR pattern generator, triple-mode address generator, SRAM under test, XOR comparator, response analyzer, and fault register. The controller exposes bist_done, bist_pass, fault_count, first_fault_address, and fault_type outputs.

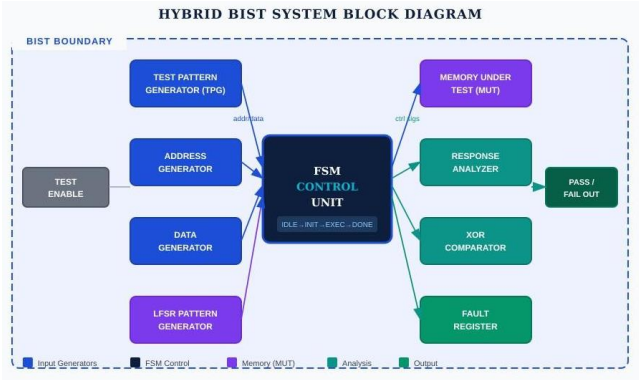


Fig. 1. Hybrid BIST System Architecture

The complete operation can be summarized as eight coordinated steps. First, test_enable disconnects the functional interface and initializes counters, seeds, and diagnostic registers. Second, deterministic backgrounds are generated. Third, ascending writes and descending reads sensitize static and coupling faults. Fourth, a programmable retention hold tests leakage-sensitive cells. Fifth, the controller switches to pseudorandom data and noncontiguous addresses. Sixth, every response is compared immediately rather than compacted only at the end. Seventh, mismatches are classified and logged with their physical address. Finally, the controller restores the normal interface and publishes pass/fail, completion, and diagnostic status. This ordered flow ensures that each phase has a defined fault objective and that a later probabilistic phase does not erase evidence collected during deterministic testing.

B) Deterministic and Pseudorandom Test Sequence

The deterministic phase uses structured backgrounds. Pass 0 writes 00h and verifies zeros; Pass 1 writes FFh and verifies ones, sensitizing both stuck-at polarities and transition behavior. Pass 2 applies AA/55h checkerboards to place adjacent bits in opposite states and expose inversion or idempotent coupling. Address traversal alternates between ascending and descending order so that each cell acts as aggressor and victim relative to its neighbors. A programmable hold interval is inserted after a write background for retention-fault screening.

The pseudorandom phase uses the primitive polynomial $P(x) = x^8 + x^6 + x^5 + x^4 + 1$. A nonzero seed produces 255 data patterns before repetition. A second LFSR with a different feedback relation drives random address traversal, reducing correlation between data and location. This phase increases the probability of activating neighborhood pattern-sensitive and timing-dependent faults that are difficult to cover with a short deterministic sequence. The controller interleaves the two modes without releasing the memory to functional operation, preserving a single test session and a single diagnostic record.

C) Response Analysis and Fault Localization

For a data width W, the response analyzer forms the editable error relation E = Dread XOR Dexpected and mismatch M = OR(E[W−1:0]). A zero error vector leaves bist_pass asserted. A nonzero vector pulses log_fault, increments the fault counter, and captures the current address. A one-hot error is classified as a likely single-bit stuck-at or transition fault; AA/55h-like disagreement during the checkerboard pass indicates coupling behavior; a full-word disagreement during a unique-address test suggests a decoder or wrong-cell selection fault. The classification is diagnostic rather than a replacement for transistor-level failure analysis, but it substantially improves repair and debug compared with a single final MISR signature.

D) RTL Top-Level Interconnection

Figure 2 shows the top-level RTL interconnection of the Hybrid BIST design. The bist_controller receives bist_en, clk, and active-low rst_n and drives the SRAM through mem_addr[7:0], mem_wdata[7:0], mem_we, and mem_re. It also provides expected_data[7:0] to the response_analyzer. The synchronous SRAM returns mem_rdata[7:0], which is compared with the expected word to generate err_vector[7:0]. When a mismatch is detected, log_fault and the current memory address are supplied to the fault_register. The register stores fault_addr[7:0], increments fault_cnt[7:0], and reports fault_type[2:0], while the controller publishes bist_done and bist_pass. This modular connection keeps control, memory access, comparison, and diagnostic storage separate, making the data path easy to trace during simulation and preserving fault-location information for later repair analysis.

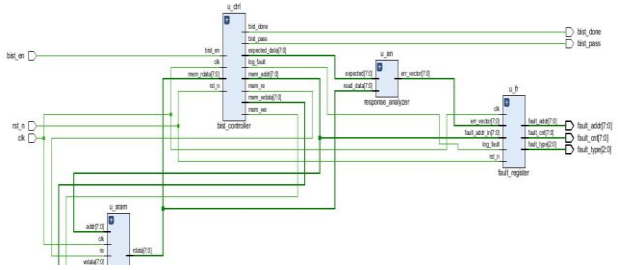


Fig. 2. RTL schematic of the Hybrid Memory Logic BIST top-level design.

E) Power-Aware Control

Dynamic power approximately follows $P_{dyn} = \alpha C L V^2 f$, where α is switching activity. During IDLE and retention hold, clock-enable signals freeze the pattern and address registers, and operand isolation prevents unnecessary data-bus transitions. Only the controller and hold counter remain active during the retention interval. These controls reduce test-mode activity while preserving the full 200 MHz at-speed phase needed for dynamic-fault screening.

Table 2. Targeted Fault Models and Test Mechanisms

Fault	Primary mechanism	Reported coverage
SAF	00/FF write-read backgrounds	100.0%
TF	0→1 and 1→0 transitions	99.2%
CF	Checkerboard, opposite traversal	98.6%
ADF	Noncontiguous address sequence	97.5%
DRF	Write, hold, read	95.1%
RWF	Interleaved read-after-write	98.2%
NPSF	LFSR neighborhood patterns	92.8%
Dynamic	At-speed LFSR stress	89.6%

F) RTL Implementation and Verification

The synthesisable implementation is partitioned into bist_controller.v, tpg.v, addr_gen.v, response_analyzer.v, fault_register.v, and a 256 × 8 synchronous SRAM model. The controller is a Moore FSM with IDLE, INIT, MARCH_WRITE, MARCH_READ, and DONE states. One-hot state realization was selected to reduce decode depth and simplify control timing. The address width and data width are parameters, so the same modules can target larger memories after adapting cycle counters and the selected FPGA RAM primitive.

The SRAM is inferred as one RAMB18E1 block on the XC7A100T. Write operations update the selected byte on the active clock edge; read data are registered to model synchronous block RAM behavior. The testbench runs at 200 MHz with a 5 ns period and checks state order, terminal-address transitions, read latency, pass/fail behavior, and captured diagnostic fields. Fault injection is performed with hierarchical force/release operations so a specific storage bit or interaction can be altered without modifying the synthesizable design.

Table 3. FSM Control States

State	Main action	Transition condition
IDLE	Normal interface; wait	bist_en = 1
INIT	Clear counters and logs	Unconditional
MARCH_WRITE	Write selected background	Last ascending address
MARCH_READ	Read, compare, and log	Last descending address
DONE	Publish result and release	Return to IDLE

G) Evaluation Metrics

Coverage is computed from injected and detected instances as $C = (N_{det}/N_{inj}) \times 100\%$. Test time is $T = N_{cycles}/f_{clk}$. Resource overhead is reported from post-synthesis utilization, and timing closure is checked against a 5 ns clock constraint. Power values are the project-reported estimates for the compared architectures. To avoid overstating results, count totals were independently reconciled before inclusion in this paper.

H) Parameterization and Reuse

Address width, data width, LFSR seed, retention count, and selected test phases are compile-time or configuration parameters. Scaling the memory changes the terminal-count logic but does not change the response relation or diagnostic interface. For word-oriented memories, the deterministic generator can cycle through additional data backgrounds so

intra-word coupling is not hidden. In a multi-memory SoC, a top-level scheduler can instantiate one controller per memory or share a programmable controller through IEEE 1500 wrappers. Shared control reduces area, whereas parallel controllers reduce total startup time; the appropriate choice depends on the power budget and the maximum allowed test interval.

IV. EXPERIMENTAL RESULTS

A) Simulation Output Results

The fault-free scenario verifies that the controller traverses IDLE → INIT → MARCH_WRITE → MARCH_READ → DONE, asserts bist_done for the intended completion interval, keeps bist_pass high, and records zero faults. In the stuck-at-0 scenario, address 42h bit 3 is forced low. When FFh is expected, the SRAM returns F7h, producing error vector 08h. The analyzer detects the mismatch, records address 42h, classifies a single-bit fault, and drives bist_pass low. The phase report also describes a coupling injection between addresses 80h and 81h to confirm checkerboard sensitivity.

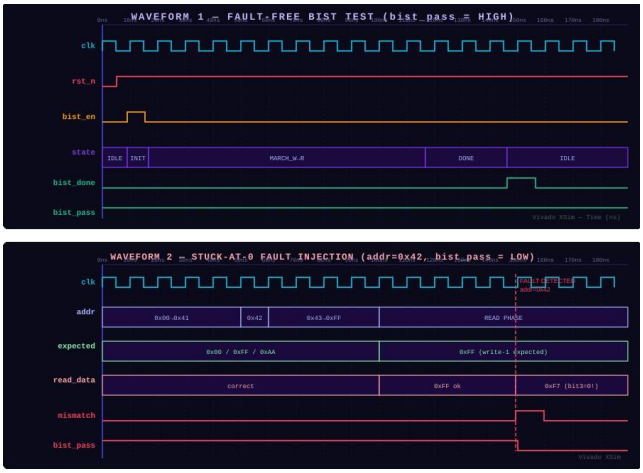


Fig. 3. Vivado XSim Results for Fault-Free and Stuck-at-0 Tests

B) FPGA Resource and Timing Results

Table 4. Artix-7 XC7A100T Synthesis Results

Resource	Used	Available	Utilization
LUTs	187	63,400	0.30%
Flip-flops	94	126,800	0.07%
RAMB18E1	1	270	0.37%
DSP slices	0	240	0.00%
I/O pins	23	210	10.95%

The control and diagnostic logic occupies only a small fraction of the FPGA. The report expresses the BIST logic as 3.2% of the memory peripheral-logic equivalent, which remains below the stated 5% design target. The critical paths from the state register, LFSR register, and response comparator all meet the 5 ns constraint. Positive slack was reported for each listed path, confirming 200 MHz operation in the evaluated design.

C) Fault-Injection Coverage

The detailed phase-report table injects 512 instances in each of eight categories. Adding its category rows gives 3,948 detections from 4,096 injections, not the 4,048 total printed in the original total row. The reproducible aggregate is therefore 96.4%. This correction does not change the qualitative result: the hybrid architecture is strongest for deterministic static faults and remains comparatively weaker for neighborhood and dynamic faults, where coverage is probabilistic and strongly dependent on sequence length, seed, timing, and the behavioral fault model.

Stuck-at coverage is complete because both stored polarities are written and observed. The small transition and coupling misses indicate that some injected behaviors require a longer or differently ordered sensitization sequence. Address-decoder coverage benefits from random traversal but can still miss equivalent decoder behaviors that produce the expected data. Retention coverage depends on the chosen hold interval, while NPSF and dynamic coverage increase as the LFSR phase is lengthened or repeated with additional seeds. These trends suggest that future optimization should allocate extra cycles specifically to the weakest categories instead of extending every phase uniformly.

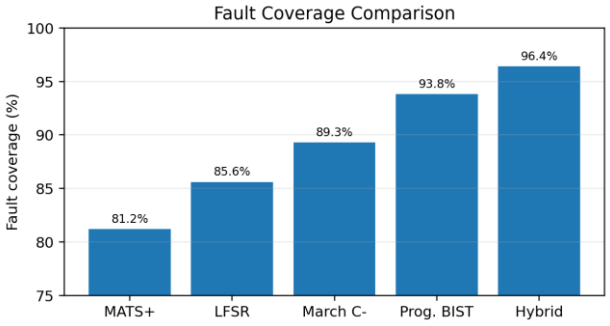


Fig. 4. Fault-Coverage Comparison

D) Comparative Performance

Table 5. Performance Comparison

Architecture	Coverage	Time	Power / Area
External ATE	~55%	~120 s	External / 0%
MATS+	81.2%	31.5 ms	18.7 mW / 1.8%
March C-	89.3%	65.5 ms	16.1 mW / 2.4%
LFSR BIST	85.6%	29.0 ms	14.3 mW / 1.5%
Programmable	93.8%	57.0 ms	15.9 mW / 3.8%
Hybrid	96.4%	42.0 ms	12.4 mW / 3.2%

Relative to the programmable baseline, the hybrid design reduces 11.4 million cycles to 8.4 million cycles and 57.0 ms to 42.0 ms. Both changes correspond to a 26.3% reduction, rather than the 36% claim appearing elsewhere in the phase report. Dynamic power decreases from 15.9 mW to 12.4 mW, a 22.0% reduction, and area decreases by 0.6 percentage points. The hybrid result therefore represents a balanced improvement: coverage is higher while power, area, and test time remain moderate. LFSR-only testing is faster and smaller but does not provide the same deterministic guarantees or fault localization.

E) Interpretation and Limitations

Behavioral fault injection is valuable for verifying controller observability, but it is not equivalent to silicon characterization. Coverage depends on how accurately each injected model represents resistive opens, weak transitions, decoder analog behavior, and timing margins. The fault classifier can label observable error patterns, yet several physical defects can produce the same digital syndrome. Furthermore, FPGA block RAM differs electrically from a six-transistor ASIC SRAM; therefore the prototype validates architecture and timing, not transistor-level defect sensitivity or absolute ASIC power.

The 5 ms retention window requires one million cycles at 200 MHz. In a production design this interval should use a low-power hold state or divided clock so retention testing

does not dominate active energy. A larger memory also requires attention to test interruption, memory partitioning, and background scrubbing. Multiple simultaneous faults may overflow a small fault register, so practical repair analysis should store a bounded list of addresses or stream events to a repair controller.

F) Safety, Deployment, and Future Extensions

For automotive, aerospace, industrial, and medical SoCs, BIST is only one part of the safety mechanism. A production safety case must combine memory BIST with error-correcting codes, parity, periodic scheduling, watchdog supervision, diagnostic communication, and a defined response after failure. The test interval must be shorter than the fault-tolerant time interval, and destructive tests require data backup, memory partitioning, or execution during startup and maintenance windows. IEEE 1500 wrappers or an SoC test-access network can coordinate multiple memory instances, while IEEE 1149.1 supports board-level control and observation [16]–[18].

A practical deployment can divide memory into banks and test one bank while software executes from another. The operating system or safety monitor saves live contents, grants the BIST controller exclusive access, runs the selected phase, restores data, and records the result in a diagnostic log. A failure may trigger ECC scrubbing, use of a spare row, isolation of the bank, a controlled reset, or transition to a safe state. The location-aware register is therefore important: pass/fail alone indicates that a defect exists, whereas an address and syndrome support repair, trend monitoring, and evidence that a recurring fault has been contained.

Future work should implement built-in redundancy analysis so logged addresses are mapped automatically to spare rows or columns. ASIC synthesis and place-and-route are needed to measure real cell-area overhead, clock-tree power, and PVT timing. Additional work can evaluate March SS or linked-fault sequences, configurable LFSR polynomials, multi-seed pseudorandom phases, and adaptive sequence selection driven by manufacturing statistics. Emerging memories such as STT-MRAM, RRAM, PCM, and 3-D NAND require technology-specific fault models beyond the SRAM-centric set used here [23]–[25].

A reusable IP generator is another practical extension. Given address width, data width, target faults, clock rate, and maximum test time, a script could emit parameterized RTL, a verification testbench, XDC constraints, and a fault-injection campaign. Automated parsing of VCD or simulation logs would calculate coverage, confidence intervals, cycle counts, and regression differences. Such tooling would turn the present architecture into a repeatable design-and-validation flow rather than a single fixed prototype.

G) Structured Fault-Injection Campaign

A reliable BIST result requires more than observing one successful waveform. The verification campaign should begin with a fault-free golden run for every supported memory configuration. The golden run establishes the expected state sequence, address order, pattern values, read latency, completion cycle, and final signature of diagnostic outputs. Each fault-injection run then changes only one controlled property of the memory model while all seeds and configuration registers remain fixed. This isolation permits a direct relationship between the injected behavior and the observed syndrome.

For stuck-at and transition faults, every address and bit position should be exercised in both polarities. Coupling campaigns should enumerate aggressor-victim pairs and distinguish inversion, idempotent, and state coupling. Decoder campaigns should model no-selection, multiple-selection, and wrong-selection cases. Retention faults require several leakage thresholds around the selected hold interval, while read/write disturbances require checking both the accessed cell and its adjacent victims. NPSF and dynamic faults should be repeated with several LFSR seeds because one seed may not generate the sensitizing neighborhood or temporal sequence.

The testbench should use assertions for protocol correctness in addition to scoreboarding the final result. Useful assertions include: test control never drives the memory when test_enable is low; write and read enables are never asserted illegally together; the address changes only in the direction selected by the current phase; bist_done is not raised before the terminal count; a mismatch always causes a diagnostic log event; and a fault-free run never lowers bist_pass. Assertions detect controller defects that could otherwise be hidden by a coincidentally correct final flag.

H) Diagnostic Evidence and Repair Interface

When a mismatch occurs, the minimum useful diagnostic record is the failing address, test phase, expected word, observed word, and error vector. Storing only the first fault is adequate for a small demonstrator, but production redundancy repair usually needs multiple unique row and column addresses. A bounded FIFO can retain the first K events, while a saturating counter indicates that additional faults occurred after the FIFO became full. Duplicate suppression prevents the same defective bit from consuming every entry during repeated backgrounds.

A built-in redundancy-analysis unit can convert these events into spare-row and spare-column assignments. The repair algorithm first groups failures by row and column, selects a set of spares that covers the maximum number of failures, programs fuse or nonvolatile repair data, and reruns BIST to confirm repair. If repair resources are insufficient, the controller reports an unrepairable status. This closed loop changes BIST from a screening mechanism into a yield-improvement mechanism and is particularly valuable for large embedded SRAM macros.

Table 6. Recommended Validation Evidence

Evidence	Purpose	Acceptance criterion
Golden regression	Confirm legal sequence	All configurations pass with zero logs
Directed injections	Verify each fault class	Expected address and syndrome captured
Multi-seed LFSR runs	Measure probabilistic stability	Coverage variation reported
Timing reports	Verify at-speed operation	No setup or hold violation
Power analysis	Check test-mode envelope	Within package and thermal budget
Repair rerun	Validate redundancy mapping	Repaired array passes complete BIST

I) Reproducibility and Reporting

Every published result should be traceable to a configuration record containing RTL revision, simulator version, FPGA part, clock constraint, memory dimensions, March sequence, LFSR polynomials and seeds, retention count, fault-model implementation, number of injected cases, and the script used to aggregate detections. Simulation

waveforms alone do not establish coverage; machine-readable event logs and a deterministic post-processing script are needed so another researcher can reproduce the totals. Synthesis and power reports should likewise identify tool settings, activity assumptions, and whether the quoted area includes the memory block or only BIST peripheral logic.

Coverage should be reported by fault category and with a count-consistent total. For pseudorandom phases, several seeds should be evaluated and the mean, minimum, and confidence interval should be given. Test time should distinguish active high-frequency cycles from long low-power retention waits. When comparing architectures, memory size, data width, clock rate, fault set, and power-estimation method must be held constant. These reporting rules prevent a faster but less comprehensive test from appearing superior merely because it targets fewer fault models.

J) ASIC Migration and Physical Validation

The FPGA prototype demonstrates digital correctness, but ASIC migration introduces clock-tree insertion, interconnect parasitics, voltage drop, memory-macro timing models, and process variation. The BIST controller should therefore be synthesized with the target memory compiler views, followed by static timing analysis across slow, typical, and fast process-voltage-temperature corners. Gate-level simulation with back-annotated delays can reveal control hazards that are absent from RTL simulation. Power analysis should use switching activity captured during the most aggressive test phase rather than a generic toggle factor.

Silicon validation should include shmoo testing across voltage and frequency, temperature sweeps, retention-time characterization, and deliberate stress patterns. Correlation between BIST diagnoses and physical failure analysis is necessary to validate the fault classifier. The final production flow should also test the BIST hardware itself through scan, controller self-checks, duplicated status logic, or periodic comparison against known responses. A defective test controller can otherwise create a false pass, which is more dangerous than a detected memory failure in a safety-related system.

K) Coverage, Time, and Hardware Balance

The central engineering decision in a hybrid BIST is how many deterministic elements and pseudorandom patterns to execute. Deterministic March operations provide predictable coverage but add a fixed number of operations per address. Pseudorandom testing has small control hardware, yet its coverage improves gradually with sequence length. A practical controller should therefore expose phase-enable bits and cycle limits rather than hard-code one universal schedule. A low-cost consumer device can run a short startup sequence, whereas a safety controller can execute the complete deterministic set at boot and schedule additional pseudorandom stress during maintenance windows.

L) Memory Size and Data-Width Scaling

For a memory containing 2^A words of W bits, the address generator requires A state bits, the comparator requires W XOR gates, and deterministic test time grows linearly with the number of words and March operations. Wider words improve test throughput but increase simultaneous switching and can hide intra-word coupling unless several data backgrounds are used. Walking-one, walking-zero, checkerboard, inverse-checkerboard, and byte-lane backgrounds can be selected according to the physical organization of the target macro. Very large memories may be partitioned into banks so one bank is tested at a time, limiting peak power and fault-log bandwidth.

The retention count must be interpreted in physical time rather than only cycles. If the test clock changes, the count should be recalculated so the hold interval remains constant. Temperature and supply voltage also affect leakage; consequently a production retention test may use different intervals at wafer test, package test, and in-field operation. The controller should allow a bypass or shortened interval for fast manufacturing screens and a longer interval for periodic diagnostic maintenance.

M) Diagnostic Storage and Result Reporting

A first-fault register is sufficient for demonstrating fault location, but the required diagnostic depth depends on the repair strategy. If the memory has two spare rows and two spare columns, retaining only four failures can be enough for a simple repair trial, while yield learning may require hundreds of syndromes. A configurable FIFO, row/column summary counters, and a sticky overflow flag provide a scalable compromise. The external interface should report test configuration, start and completion times, pass/fail status, total fault count, overflow state, and the list of retained events.

Table 7. Example Configuration Profiles

Profile	Deterministic phase	LFSR phase	Diagnostic policy
Fast production screen	SAF/TF core March	One seed, short run	First fault + count
Extended production	Full static fault set	Multiple seeds	FIFO for repair analysis
Power-on self-test	Non-destructive subset	Optional	Persistent pass/fail log
Safety maintenance	Banked full test	Scheduled stress	Timestamped events + safe action

N) Recommended Integration Practice

Integration should begin with a memory-interface contract that defines read latency, write behavior, byte enables, reset values, and ownership arbitration. The BIST wrapper must prevent simultaneous functional and test access and should acknowledge when the memory is safely isolated. Clock-domain crossings are avoided when possible; otherwise start, done, and diagnostic signals require synchronizers or a request-acknowledge handshake. Reset behavior must be verified so an interrupted test cannot leave the memory interface permanently disconnected.

Before release, the same configuration should be tested in RTL simulation, post-synthesis simulation, static timing analysis, and on-board hardware. The hardware run should compare test duration and diagnostic outputs with simulation, while embedded logic analyzers can capture state and mismatch events. Regression tests should be repeated after any change to memory dimensions, compiler options, clock constraints, or FPGA/ASIC technology. These practices preserve the parameterized design advantage without allowing configuration changes to invalidate fault coverage silently.

V. CONCLUSION

This paper presented a Hybrid Memory Logic Based BIST architecture that integrates deterministic March testing with LFSR-based pseudorandom stress under one FSM. The design supports ascending, descending, and random address

traversal, bitwise response comparison, fault-type classification, and address-level logging. Verilog simulation confirms correct fault-free behavior and detection of an injected stuck-at fault, while Artix-7 synthesis demonstrates a compact implementation using 187 LUTs, 94 flip-flops, one BRAM, and no DSP blocks at 200 MHz. The project-reported test time is 42.0 ms, dynamic power is 12.4 mW, and peripheral-logic overhead is 3.2%. Reconciliation of the detailed fault-injection entries yields 96.4% overall coverage. The results show that hybridization can improve fault breadth without the repeated-pass penalty of a fully programmable architecture, while preserving the diagnostic information required for repair and in-field monitoring.

REFERENCES

[1] A. J. van de Goor, Testing Semiconductor Memories: Theory and Practice. Chichester, U.K.: Wiley, 1991.
[2] D. S. Suk and S. M. Reddy, "A March test for functional faults in semiconductor random-access memories," IEEE Trans. Comput., vol. C-30, no. 12, pp. 982–985, Dec. 1981.
[3] P. H. Bardell, W. H. McAnney, and J. Savir, Built-In Test for VLSI: Pseudorandom Techniques. New York, NY, USA: Wiley, 1987.
[4] M. Renovell, J. Figueras, and Y. Zorian, "RAM-based programmable BIST for memory testing," in Proc. IEEE Int. Workshop Memory Technology, Design and Testing, 1997, pp. 15–20.
[5] M. A. Franklin, K. K. Saluja, and K. Kinoshita, "Built-in self-testing of random access memories," Computer, vol. 23, no. 3, pp. 45–56, Mar. 1990.
[6] R. Dekker, F. Beenker, and L. Thijssen, "A realistic fault model and test algorithms for static random access memories," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 9, no. 6, pp. 567–572, Jun. 1990.
[7] Z. Al-Ars and A. J. van de Goor, "Static and dynamic behavior of memory cell array opens and shorts in embedded DRAMs," in Proc. Design, Automation and Test in Europe, 2000, pp. 496–503.
[8] S. Hamdioui, Z. Al-Ars, and A. J. van de Goor, "March SS: A test for all static simple RAM faults," in Proc. IEEE Int. Workshop Memory Technology, Design and Testing, 2002, pp. 95–100.
[9] R. Nair, S. M. Thatte, and J. A. Abraham, "Efficient algorithms for testing semiconductor random-access memories," IEEE Trans. Comput., vol. C-27, no. 6, pp. 572–576, Jun. 1978.
[10] P. Bernardi, M. Grosso, and M. Rebaudengo, "March-based algorithms for embedded-memory BIST in system-on-chip designs," in Proc. IEEE European Test Symp., 2006, pp. 109–114.
[11] S. W. Golomb, Shift Register Sequences, rev. ed. Laguna Hills, CA, USA: Aegean Park Press, 1982.
[12] C. E. Stroud, A Designer's Guide to Built-In Self-Test. Boston, MA, USA: Kluwer, 2002.
[13] Y. Zorian, "A distributed BIST control scheme for complex VLSI devices," in Proc. IEEE VLSI Test Symp., 1993, pp. 4–9.
[14] M. Nicolaidis, B. Courtois, and C. Dussault, "Efficient BIST design for embedded SRAMs," in Proc. IEEE Int. Conf. Computer Design, 2000.
[15] N. A. Touba, "Survey of test vector compression techniques," IEEE Design Test Comput., vol. 23, no. 4, pp. 294–303, Jul.–Aug. 2006.
[16] ISO 26262, Road Vehicles—Functional Safety, 2nd ed. Geneva, Switzerland: International Organization for Standardization, 2018.
[17] IEEE Standard Testability Method for Embedded Core-Based Integrated Circuits, IEEE Std 1500-2005, 2005.
[18] IEEE Standard for Test Access Port and Boundary-Scan Architecture, IEEE Std 1149.1-2013, 2013.
[19] R. Bearden and C. Stroud, "FPGA-based memory BIST with March algorithm support," in Proc. IEEE Midwest Symp. Circuits Syst., 2006, pp. 271–274.
[20] AMD, 7 Series FPGAs Memory Resources User Guide, UG473, San Jose, CA, USA, 2023.
[21] AMD, Vivado Design Suite User Guide: Logic Simulation, UG900, San Jose, CA, USA, 2023.
[22] AMD, Vivado Design Suite User Guide: Synthesis, UG901, San Jose, CA, USA, 2023.
[23] S. Hamdioui, R. Wadsworth, J. D. Reyes, and A. J. van de Goor, "Importance of dynamic faults for new SRAM technologies," in Proc. IEEE European Test Workshop, 2003, pp. 29–34.
[24] M. L. Bushnell and V. D. Agrawal, Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits. Boston, MA, USA: Kluwer, 2000.
[25] M. Abramovici, M. A. Breuer, and A. D. Friedman, Digital Systems Testing and Testable Design. New York, NY, USA: IEEE Press, 1990.
[26] A. J. van de Goor and G. N. Gaydadjiev, "March LR: A test for realistic linked faults," in Proc. IEEE VLSI Test Symp., 2005, pp. 272–280.
[27] J. F. Ziegler et al., "IBM experiments in soft fails in computer electronics (1978–1994)," IBM J. Res. Develop., vol. 40, no. 1, pp. 3–18, Jan. 1996.
[28] AMD, Vivado Design Suite User Guide: Using Constraints, UG903, San Jose, CA, USA, 2023.
[29] D. K. Pradhan, Ed., Fault-Tolerant Computer System Design. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.
[30] D. Gizopoulos, Ed., Advances in Electronic Testing: Challenges and Methodologies. Dordrecht, The Netherlands: Springer, 2006.