International Journal of Electrical and Computer Engineering (IJECE)
Vol. 14, No. 2, April 2024, pp. 1406~1423
ISSN: 2088-8708, DOI: 10.11591/ijece.v14i2.pp1406-1423

Testing nanometer memories: a review of architectures, applications, and challenges

Vijay Sontakke, Delsikreo Atchina
Intel Corporation, Allentown, United States

Article Info

Article history:

Received Aug 31, 2023
Revised Dec 9, 2023
Accepted Jan 5, 2024

Keywords:

3D memory test
Built-in self test
In-system test
Linear feedback shift register
March algorithm
Power-on self test

ABSTRACT

Newer defects in memories arising from shrinking manufacturing technologies demand improved memory testing methodologies. The percentage of memories on chips continues to rise. With shrinking technologies (10 nm up to 1.8 nm), the structure of memories is becoming denser. Due to the dense structure and significant portion of a chip, the nanometer memories are highly susceptible to defects. High-frequency specifications, the complexity of internal connections, and the process variation due to newer manufacturing technology further increased the probability of the physical failure of memories to a great extent. Memories need to be defect-free for the chip to operate successfully. Therefore, testing embedded memories has become crucial and is taking significant test costs. Researchers have proposed multiple approaches considering these factors to test the nanometer memories. They include using new fault models, march algorithms, memory built-in self-test (MBIST) architectures, and validation strategies. This paper surveys the methodologies presented in recent times. It discusses the core principles used in them, along with benefits. Finally, it discusses key opens in each and offers the scope for future research.

Corresponding Author:

Vijay Sontakke
Intel Corporation
1110 American Pkwy, Allentown, PA, 18109, United States
Email: vijay.sontakke@intel.com

1. INTRODUCTION

Semiconductor memories are widely used in every computing system, including sensors, desktops, and servers [1]. In the last decade, data-intensive applications started using huge amounts of memory. Since the number of I/O pins is limited, external testing of embedded memory cores is difficult. Memory built-in self-test (MBIST) has been proven to be a reliable and efficient method of testing these memories. They execute multiple algorithms to detect various types of defects. Memory tests are designed to check the functionality of memory cores, address uniqueness, decoder speed, cell coupling, and data sensitivity. Memory testing requires specific pattern sequences to exercise.

Defects in memories are one of the primary defects found after fabrication. So, these defects in memories always necessitated to have repair capabilities to improve yield. Several techniques are being used to add repair and correction capabilities for memories. Our previous work [2] presents a comprehensive study of the techniques.

The growing use of electronics in safety-critical applications increased demand for more reliable operation of system-on-chip (SOC). The need for higher safety and reliability specifications for automotive usage gave birth to the ISO 26262 standard [3]. It necessitates testing components not only during production but also in the field. A few acronyms are frequently used in the paper. Table 1 gives full forms of them. The table also contains full forms of a few abbreviations being used.

Table 1. Abbreviations and acronyms

Abbreviation	Full form	Abbreviation	Full form
MBIST	Memory built-in self test	SOC	System on chip
PSF	Pattern sensitive faults	IRF	Incorrect read fault
POST	Power-on self test	IST	In-system test
WDF	Write disturb faults	ATPG	Automatic test pattern generation
RDF	Read disturb faults	ATE	Automatic test equipment
PVT	Process, voltage, and temperature	MISR	Multiple input signature register

Academia and industry have proposed many techniques to test nanometer memories, including static random-access memory (SRAM), read only memory (ROM), and dynamic random-access memory (DRAM). This paper describes the principles, algorithms, and architecture details used in them. It classifies the techniques, discusses the advantages and disadvantages of each type of those techniques, and enables selection for particular usage.

The organization of the remaining sections of this paper is as follows. Section 2 gives a background of memory testing. Then, section 3 classifies techniques based on the principle used and its application. Details of the techniques for memory testing are discussed in sections 4, 5, and 6. Finally, section 7 discusses the pros and cons of the techniques, while section 8 concludes the paper.

2. BACKGROUND

Metallization shorting and capacitive coupling are two common manufacturing defects in memories. These defects could result in a single cell or multiple cells of the memory array being faulty. Multiple cell faults could also be due to the operation of other cells. While single-cell faults are straightforward to detect, multiple-cell faults require a series of operations. Cell array, read-write logic, and address decoder should be tested for correct operation [4]. Following are a few commonly found faults:

a. Stuck-at faults (SAF): This fault is like a fault for the logic cell. The cell value is either 0 or 1 and cannot be altered. Each cell should be written with 0 and 1 and read back to detect this fault.
b. Transition faults (TF): This fails to change the value after writing the cell. The cells do not respond to rising (0 to 1) or falling (1 to 0) transition. The fault could be detected by writing and reading the cell successively with 0 and 1 (for rising fault detection) or with 1 and 0 (for falling fault detection).
c. Coupling faults (CF): This fault results in a cell giving the wrong value due to the effect of other cells' values or operations being performed on them. Two or more adjacent cells alter the value in a particular cell due to the fault.
d. Data retention fault (DRF): This fault results in losing value in a cell after a while. The loss usually occurs when the pull-up circuit malfunctions and leakage current results in losing the capacitance value. A simple writing operation, followed by reading after a certain period, detects this fault. Adjacent cells are written with opposite values to exacerbate the effect,
e. Address decoder fault (ADF): This fault results in wrong address decoding values. They are classified into four types: i) a cell is not accessible by any address, ii) multiple addresses access the same cell, iii) multiple cells are accessed by one address, and iv) some address accesses no cell. These faults are detected when the memory array is accessed simultaneously in ascending and descending address order for 0 and 1 [4].

2.1. Testing algorithms

In order to check if the memory array contains faults, a series of write and read operations need to be performed. The March algorithms are the most commonly used. They perform a sequence of write and read operations by traversing through the entire address space in ascending and descending address order. It completes operation on a cell before starting operation on the next cell in order. A march element (M) consists of a set of operations being performed on a cell. For example, the operation performed by the March B algorithm is described below. This algorithm performs 17 operations on each memory cell.

Algorithm: March B

$\{\updownarrow (W0);\ \uparrow (R0,W1,R1,W0,R0,W1);\ \uparrow (R1,W0,W1);\ \downarrow (R1,W0,W1,W0);\ \downarrow (R0,W1,W0)\}$

$M0$ $M1$ $M2$ $M3$ $M4$

Notations:
Up arrow (↑): traversing address space in ascending order
Down arrow (↓): traversing address space in descending order
W0: write 0, W1: write 1, R0: read 0, R1: read 1

The algorithm performs the operation by using different backgrounds of data to increase the coverage of faults [5]. First, the memory array is initialized with the background data. Then, a series of write-read operations start. Backgrounds like solid 1 or 0, column stripe, row stripe, and checkboard are commonly used. These backgrounds are shown in Figure 1.

Similar to data backgrounds, the operations are performed with different addressing schemes to excite defects in the address decoder or the cells. A few commonly used addressing schemes are:

a. Fast row: Cells are accessed by incrementing/decrementing row address lines of the address decoder.
b. Fast column: Cells are accessed by incrementing/decrementing column address lines of the address decoder.
c. Gray code: Addresses are changed by keeping Hamming distance as 1.
d. Address complement: Addresses are changed by keeping the maximum Hamming distance.

Various data backgrounds and addressing schemes contribute to covering a variety of faults. For example, for performance improvements, memories have implemented banks. Due to this, two adjacent cells represented by logical address may not physically reside next to each other. So, the addressing and data schemes must be implemented considering the physical location.

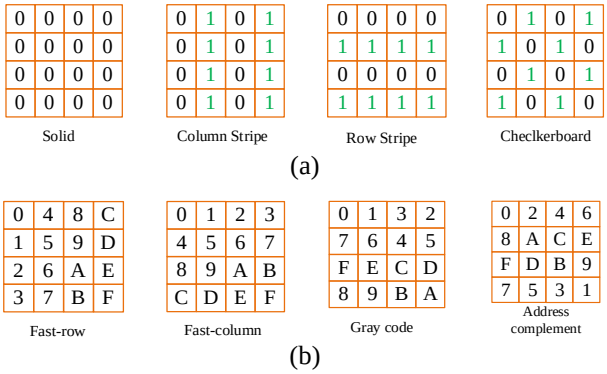

Figure 1. Commonly used data and address backgrounds (a) data backgrounds and (b) address backgrounds

2.2. MBIST architecture

A memory built-in self-test controller usually consists of an address generator, data generator, control logic, and comparator. Such a controller is shown in Figure 2. Muxes are added on address, data-in, read-write enable, and other control ports to enable paths from the built-in self-test (BIST) controller. The control block receives commands to perform read or write operations at a particular address. It controls overall operation and executes various algorithms. The comparator block checks output coming from memory with the expected value provided by the control block and asserts pass or fail signals. Various algorithms are implemented and selected during testing by configuring the control block. On a typical SOC, memories are grouped based on physical proximity. Separate controllers are used to check these groups of memories and operate in parallel to reduce test time.

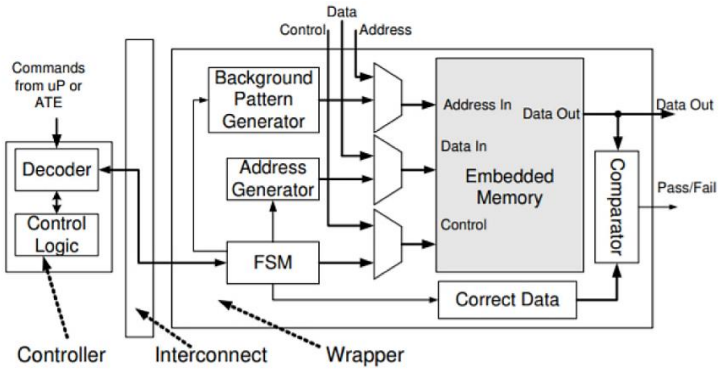

Figure 2. Memory built-in self-test controller block diagram [6]

2.3. Previous surveys on memory testing

Hamdioui *et al.* [7] presented a survey on testing embedded memories. It covered fault models for memories and various testing algorithms. It also discussed future challenges regarding reducing voltage, increasing operation frequency, and process, voltage, and temperature (PVT) variations. Another survey was presented in [8], which discussed a few MBIST architectures and algorithms. Both these surveys were presented in 2013. New generation memories have been used since then, and many new methodologies to test them were presented later. This paper presents a survey of those methodologies.

3. CLASSIFICATION

Over the years, researchers have presented various approaches to test memories. This survey covers the approaches published in recent times. They are divided into categories based on the critical parameters of testing. Tables 2, 3, and 4 present these categories.

Table 2. Testing methodologies

Sr.	Technique	References
1	3D memories testing	[9], [10]
2	In-field testing	[1], [9], [11]–[16]
3	Test scheduling	[17]–[20]
4	Standardization efforts	[21]–[23]
5	Write-through testing	[24], [25]
6	DRAM/ROM/CAM testing	[6], [16], [26]–[28]
7	Low power MBIST	[29]–[33]
8	Pipelining interface	[34]–[36]
9	Reliability testing	[37]–[39]
10	FPGA implementation of MBIST	[29], [35], [40]–[45]

Table 3. Algorithmic approaches

Sr.	Technique	References
1	New algorithms	[42], [46]–[53]
2	Algorithms for pattern sensitive faults testing	[54]–[58]
3	MBIST controllers with configurable algorithms	[34], [59], [60]
4	Algorithms combining	[61]

Table 4. Validation strategies

Sr.	Technique	References
1	Collection of diagnostic data	[62]
2	Algorithm trimming	[63]
3	Algorithm verification	[5]
4	Reduced address verification	[64]
5	Physical failure analysis	[65]

4. TESTING METHODOLOGIES

4.1. In-field testing

The use of electronics-controlled systems has increased in every industry. Safety is of utmost importance in these systems, especially in automotive, medical, space, and aviation. For the automotive industry, the ISO 26262 standard defines safety requirements. One inherent problem with using electronics in the automotive industry is using technology before it matures [1]. This problem necessitates the use of a robust testing methodology. In addition to production mode testing, systems must be checked before use to ensure error-free functioning. Also, periodic testing must be done while the system operates (i.e., mission mode) to detect and correct any spurious defects due to environmental conditions. To perform both these types of testing, testing must be enabled in the field. To support these two requirements, researchers presented a few architectures. This section summarizes them.

Im *et al.* [13] presented an architecture for automotive SOC which supports in-field testing. The architecture includes power-on self-test (POST) and in-system test (IST) controller blocks that interface with advanced peripheral bus (APB), as shown in Figure 3. These blocks help perform memory tests, repair, and logic BIST operations.

Figure 4 shows the sequence of operations being performed during in-field usage using architecture in [13]. First, the POST controller initiates a memory test during power-up and boot flow. Memories are tested in this step to ensure defect-free functional operation. The in-system test flow is later controlled by the

CPU, which runs memory built-in self-test (MBIST) and logic built-in self-test (LBIST) operations on targeted blocks.

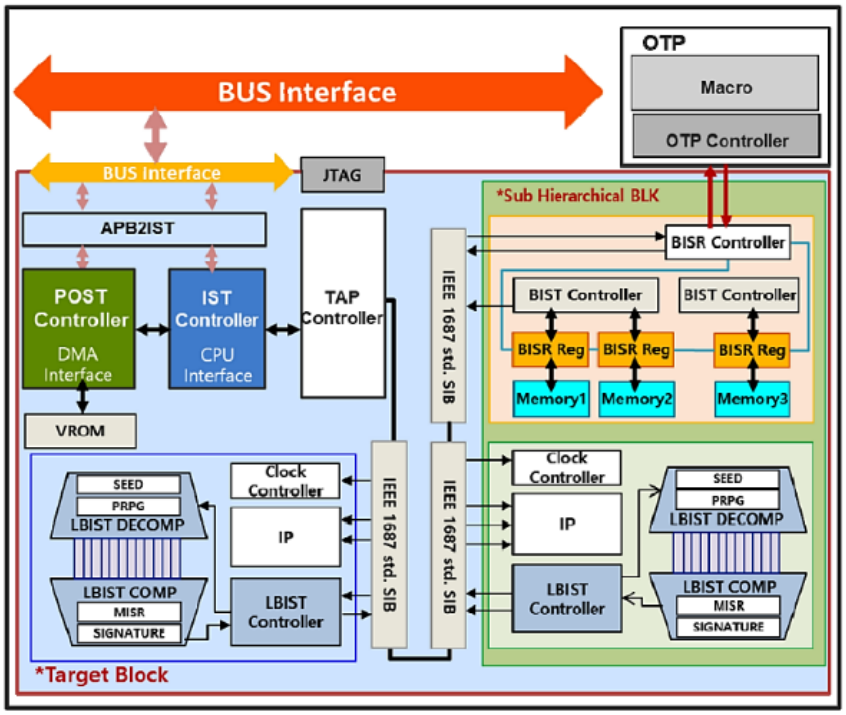

Figure 3. DFT architecture for automotive SOC [13]

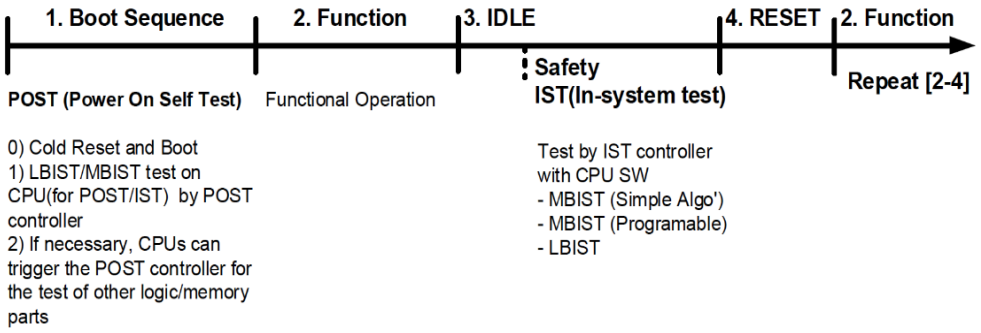

Figure 4. Testing steps during functional operation [13]

Sargsyan *et al.* [1] presented an architecture that supports in-field memory testing. It uses Synopsys STAR Memory System (SMS) for memory testing and repair. It provides a short and efficient algorithm to run during power on to check faults like stuck-at, transition, stuck-open, and read-destructive. The power-on test aims to check memories for defects in a short period. The short yet efficient algorithm helps to perform this check. During mission mode, SMS provides a test access port to control operations for periodic memory tests. It continuously monitors which memories are busy and can be selected for testing. Reserve registers are provided, which are used to store memory contents temporarily. The contents are restored after the completion of the test. Testing is done by selecting a range of addresses at a time. It also incorporates multi-bit error detection and correction codes. The correction codes handle soft errors during mission mode due to the emission of alpha particles and cosmic rays. Sargsyan [11] later presented a firmware generation methodology for systems employing SMS. He also presented a detailed structure of firmware C code for a case study.

Becker *et al.* [12] presented a methodology for online testing of memories in an ARM processor. It uses a short, 20-cycle transaction to test memories using the MBIST controller. The transactions are separated by 2,000 to 20,000 cycles apart. It temporarily stores memory content while testing and restores it after finishing testing. It locks memory during this period. The MBIST controller also compares data with an error correction code (ECC) and writes the correct value in case of an error.

The methodology presented in [12] has a disadvantage. Locking of memory results in stalling if the processor needs access to it. Though the duration and probability are minimal, the system may experience a slight performance hit.

Jagannadha *et al.* [15] presented another architecture to check field defects using in-system testing (IST). The architecture could be used in autonomous drive platforms. The IST supports key-on and key-off testing. The IST controller communicates through IEEE 1500 to execute MBIST operation. In case of failure, it unloads details for debugging using diagnostic software.

Angione *et al.* presented details of the impact due to periodic in-field testing at the operating system (OS) layer [14]. OS stores data and instructions in embedded memories. Any defects in them result in non-deterministic behavior. The paper proposes to perform concurrent memory testing without impacting the requirements of Real-time-operating-systems (RTOS). Since MBIST operation is destructive, it mandates creating a copy of data in temporary storage and restoring it after testing. In addition to MBIST testing, it also uses ECC circuitry for memories to detect and correct any latent errors.

4.2. 3D memories testing

On the three-dimensional (3D) package, dies are linked via through-silicon vias (TSVs). The TSVs are further connected to respective dies using micro-bumps. Then, the memory dies are connected with logic using such TSV interconnects [66]. For the memory dies, the TSVs and interconnects also need to be checked in addition to the memory array itself. This section summarizes approaches used for testing these memories.

Harutyunyan and Zorian [9] presented a solution for testing memory dies in 3D packages. It adds memory BIST controller in the logic die and operates using a physical (PHY) interface, as shown in Figure 5. All operations on memory, including initialization, are done by the BIST controller, while the memory controller carries out PHY-initialization. The BIST controller implements a programmable test register that can be used to select different backgrounds and algorithms. For interconnects, it checks connectivity for single line and coupling faults for connections i) from the logic die to each memory die and ii) from each memory die to other memory die (intra-die).

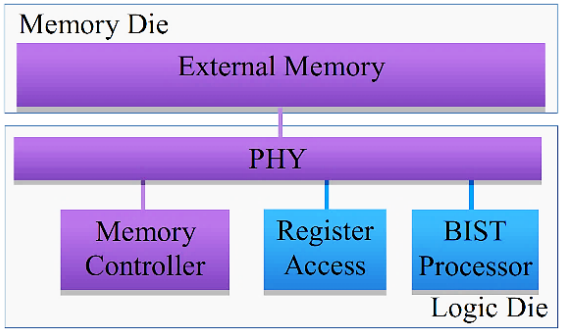

Figure 5. Testing of external memories using logic die [9]

Jani *et al*. [10] presented a DFT architecture to test interconnects in 3D integrated circuits. Die wrapper register (DWR) could be inserted around a logic core to provide isolation, which could be used for internal testing (INTEST) of the individual dies and external testing (EXTEST) for interconnects connected to them [67]. The author uses this approach, which aligns with the IEEE P1838 standard. The architecture contains a local MBIST controller in the memory dies, as shown in Figure 6. The controller checks memory arrays for defects during both pre-bond and post-bond. Interconnects between the memory die and logic die are tested by the logic die, which performs a series of write and read operations. DWR cell is added on each such interconnect. These DWRs are connected in a chain and used for applying and checking data. This architecture contains a local MBIST controller, which is easier to implement and verify than the approach in [9].

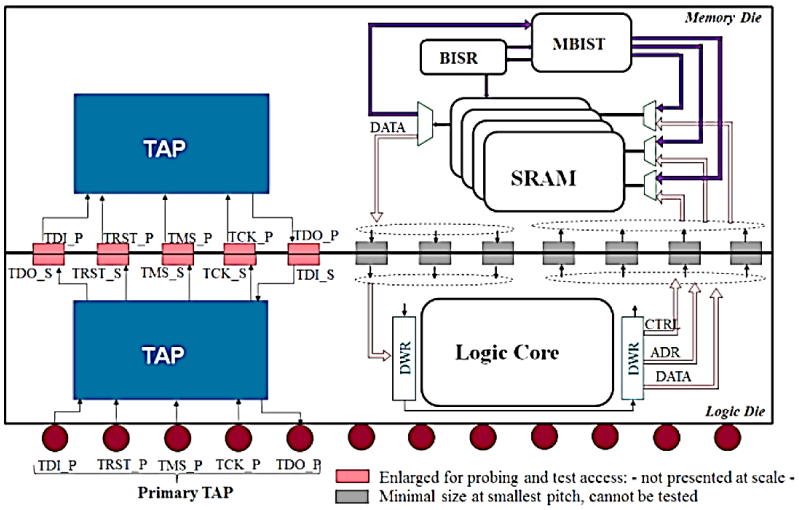

Figure 6. DFT architecture for 3D-IC [10]

4.3. Write-through testing

MBIST designing adds mux on a path to memory, which selects a signal from the BIST controller or functional logic. In most cases, designers tend to select less timing critical paths to mux with BIST data to minimize the impact of the mux delay. While this approach helps to implement BIST testing at speed, it results in missing coverage on some shadow logic. This section presents approaches to improve their coverage.

Gao *et al.* [25] presented a path delay testing methodology to test memory interface paths. It models memory with descriptions for synchronous write and asynchronous read operations. A pattern generator named CodeGen uses this model to generate tests covering paths to and from memories. Launching a transaction at memory output is achieved using preamble cycles, which initialize memory for the required value for launch. Observed values at memory inputs are taken to scan flops for checking by adding extra cycles after the capture event.

Mohammad *et al.* [24] presented a scan-based methodology to improve the coverage of memory interface logic. First, it uses a memory model which defines read and write operations. Next, it adds logic on write-enable and read-enable and a clock port to facilitate control of the operation. Then, it uses the ATPG tool to generate patterns that exercise memory, which is similar to functional operation.

4.4. Pipelining interface

A mux is traditionally added to access the MBIST controller's memory ports. In general, paths to and from memories are timing critical, and this mux further adds delay to it. Therefore, layout engineers must make a significant effort to meet such high-speed clock requirements. This section discusses the methodology which tries to solve this problem.

McLaurin and Knoth [36] presented a methodology called MBIST interface to improve the quality of memory testing. It proposes to reuse functional paths during MBIST testing. Traditional MBIST design accesses memory ports by adding mux near the actual memory array. Though it tests the memory array for rated speed, the functional path remains untested. Moving the mux to the functional control generation point allows an actual functional path to be used, and BIST operation can be done at a functional-rated speed. Description of memories and control logic needs to be available to MBIST insertion tools. The author presented details about such standardization.

Memory BIST with pipelines on memory interfaces is presented in [35]. It adds pipelines on input and output side ports of memory. In contrast, the design presented in [34] inserts the pipeline stage at the memory output signal, which goes to a comparator. The addition of a pipe stage improves the timing of the read operation. The authors showed an improved speed of BIST operation with these pipelines.

4.5. Standardization efforts

4.5.1. IEEE 1500 compliant wrapper for memories

An IEEE 1500-compliant wrapper for memories is presented in [21], which facilitates applying MBIST patterns. The wrapper does not have any finite state machine (FSM), which makes it area-efficient

and easier to implement. Instead, it contains design blocks for counting, test data generation, and shift registers. Despite not having FSM, the wrapper is fully programmable and supports the at-speed testing of memories. Wrappers presented in [68] and [69] use a serial mode of IEEE 1500 wrapper guidelines but incur high area overhead and slow testing due to their serial nature. In contrast, the wrapper proposed in [21] supports testing multiple memories in parallel.

4.5.2. IEEE 1450.6.2-2014 based memory modeling

This standard defines guidelines to describe memory functionality to assist MBIST flow. Developers from electronic design automation (EDA) tools, memory IP, and test engineering could use the models built using the guidelines. Using the same model ensures consistency among them. Arora *et al.* [22] presented details of using a core test language (CTL) description-based memory model for wrapper, MBIST controller, and pattern generation stages. It also points out challenges while using the standard guidelines. The author lists a few inconsistencies in defining error injection features and states how EDA companies are finding a workaround.

Raval *et al.* presented a flow [23] to implement MBIST design, which is IEEE 1500 and IEEE 1149.1 compliant. It uses a memory wrapper that follows IEEE 1500 standard. The BIST controller interfaces with wrapper test access port (WTAP), as shown in Figure 7. The WTAP is further controlled by a top-level test access port (TAP) controller that is IEEE 1149.1 compliant. Top-level TAP manages memory testing operations by controlling various WTAPs.

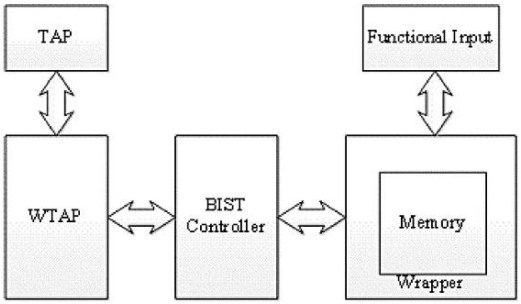

Figure 7: Memory testing using IEEE 1500 compliant blocks [23]

4.6. Test scheduling

Bhaskaran *et al.* [17] presented a design-for-test (DFT) architecture that reduces dynamic power during memory testing. Memory tests incur a lot of toggling and may lead to voltage droop and peak current limits crossing. It creates multiple groups of MBIST logic and scan logic. It adds clock gates for each of those groups. These clock gates are enabled using JTAG-controlled registers. The architecture disables the clock to all blocks except the block undergoing testing.

A similar architecture to control power dissipation is presented in [20]. Here, the author creates a control chain with bits for each MBIST controller. The output of this chain controls the *mbist_en* port instead of the clock-gating cell as in [17]. The *mbist_en* port also drives logic to enable the clock to memory and MBIST controller. Various configurations of MBIST controllers are enabled by programming this control chain.

Das and Prakash [18] presented another architecture to schedule MBIST tests considering dynamic voltage (IR) drop. As shown in Figure 8, it adds a scheduling block in the design, which runs MBIST controllers on automatic test equipment (ATE). It also finds a combination of controllers that can be run simultaneously across PVT corners. Finally, it uses a machine learning method that characterizes a few devices across various PVT corners and uses the findings to create groups of controllers that can be tested reliably during final production testing.

Zeli *et al.* [19] presented a comparative study of testing memories in parallel and sequential. Logic is required to generate multiple sets of addresses and data to test memories in parallel. It also requires multiple sets of comparators. Parallel testing saves test time but incurs significant area overhead. The author analyzes the area and test time for a design with eight memory instances. It creates the following combinations of memories and controllers. Figure 9 shows the area and test time analysis for these combinations: i) a parallel MBIST with no gangs, ii) a sequential MBIST with no gangs, iii) a parallel MBIST with vertical gang, and iv) a sequential MBIST with a horizontal gang.

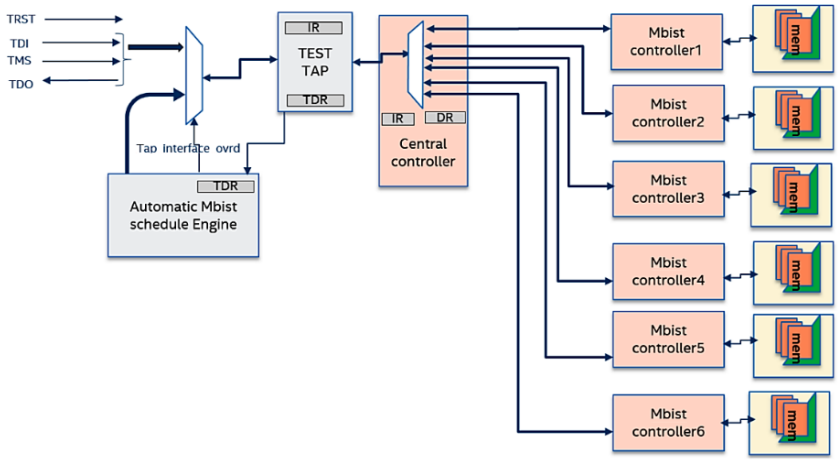

Figure 8. Architecture with automatic MBIST schedule engine [18]

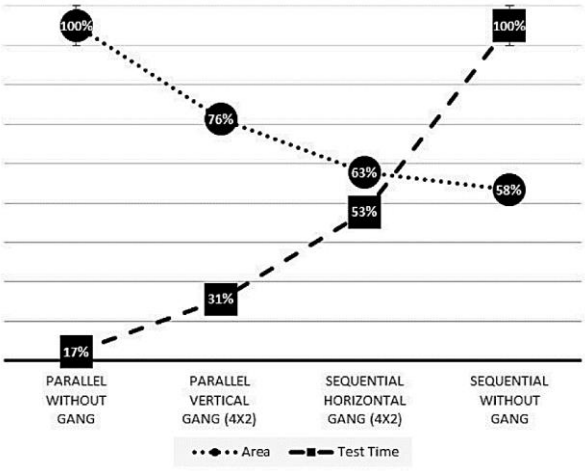

Figure 9. Area and test time with different combinations of memories [19]

4.7. DRAM/CAM/ROM testing

4.7.1. DRAM testing

Due to shrinking feature sizes, the effect of neighboring cells on the retention of data in memory bit cells is increasing. In addition, exposure to cosmic rays also affects retention capacity [70]. ECC was used historically to find weak cells prone to variable retention time (VRT). Park *et al.* [26] proposed g a BIST controller to DRAM itself. It is a direct method, and detection can be done within seconds compared to the ECC method, which takes hours to days. The author proposes a small-size BIST controller with a feature to configure retention time and implements a modified March algorithm. For 16 GB DDR4 DRAM, the area overhead due to this is only 0.051% of the total size of DRAM. Also, it saves 48% of the test time compared to other BIST controllers used for DRAM testing.

4.7.2. CAM testing

Yang *et al.* [28] presented a method to test a new 28 nm quaternary content addressable memory (CAM). Using the fault injection method, it analyzed new defect behavior and performed HSPICE simulations. The behaviors are then described in the form of the following fault models: i) stuck miss-state fault (SMSF), ii) partial-match data fault (PMDF), iii) specific data and specific compare mismatch fault (SDSCMMF), iv) pairwise search-line coupling fault (PSCF), and v) match-line under-charged fault (MLUCF).

Wu *et al.* [71] derived a new algorithm based on the steps in their study to detect these faults. The complexity of the proposed algorithm is 12N+4B+5, which is quite lower than 15N+4B+6 of the original algorithm. At the same time, it maintained the same fault coverage as the original.

4.7.3. Parallel ROM testing

ROMs were never tested in parallel as they incur significant area overhead due to the requirement of the additional backward data path and multiple input shift register (MISR) [72]. Mishra *et al.* [16] presented an MBIST architecture, as shown in Figure 10, to test multiple ROM instances in parallel. It generates one-hot enable signals for each ROM instance in a ping-pong fashion, and the clock is enabled using it. Data from multiple ROMs are routed through mux, and a single-bit stream is formed and fed to MISR with respect to the original clock source. After reading data from all addresses, the MISR output is compared with the golden signature.

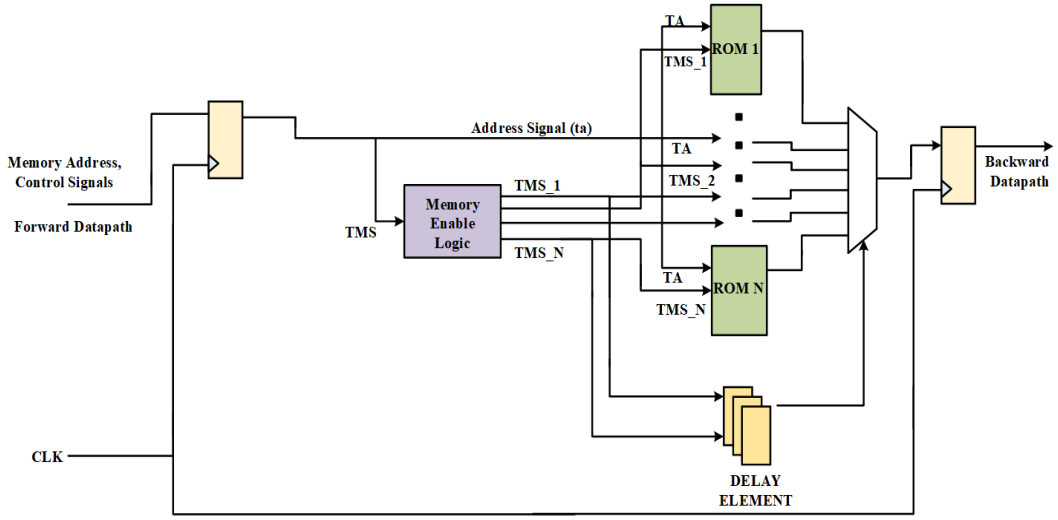

Figure 10. Architecture to test ROMs in parallel [16]

4.7.4. ROM in-field testing

In safety-critical devices, the correct functioning of ROMs at boot-up is essential. Checking its contents is prohibited as the reading operation will also provide the values to boot flow and corrupt operation. So the cyclic redundancy check (CRC) method [73] is popularly used for ROM testing at boot flow. One drawback of using CRC check is a longer run time, which can also increase the boot-up time. Mishra *et al.* [16] presented a method to check ROM using MBIST during boot flow. It copies the contents of ROM into a temporary random-access memory (RAM). Then, multiple parallel MISRs are used to check the contents of this RAM. This method can finish contents checking quickly, and boot flow remains undisturbed during this checking. The author showed that, compared to CRC-based checks, parallel MISR-based checks saved 99% of test time. The author proposes to use the approach to check ROMs at start-up, during periodic testing, or at power-down.

4.8. Low power MBIST

Many chips are designed for low-power applications like mobile phones, drones, and satellites. They use packages with small power rating specifications. Therefore, when test patterns are applied during testing, it becomes essential to limit the toggling of logic so that the resulting power consumption does not exceed the package rating and does not burn the chips and dies. Discrete logic gates and memories are the two main components of a typical chip. Many techniques have been presented to limit toggling during logic testing, and their study is presented in [74] and [75]. Since memories occupy a significant portion of the chip area, control of toggling during memory testing is also required. This section presents a summary of these techniques.

Jamal and Srihari [76] presented a study of a low-power test pattern generator using different seed algorithms. The number of transitions in bit swapping linear feedback shift register (LFSR) is 2^{n-2}, whereas, in standard LFSR, it is 2^{n-1} [77]. Another single-bit change sequence generator-based LFSR [78] has even lower transitions than these two LFSRs. Few architectures were presented that make use of such LFSRs to reduce power during MBIST operation.

Saravanan *et al.* [31] presented an MBIST architecture that uses a gray code counter and Bit Swapping LFSR for address generation. The author showed an improvement of 92.28% in power consumption compared to address generators using traditional LFSR. Another low-power MBIST is

presented in [30]. It uses a combination of modulo-counter and gray code counter for address generation. It makes use of the counters to generate reversible patterns effectively. Using this method, the author showed a reduced switching power from 0.9203 to 0.8684 mW.

A low transition address generator has been presented in [29]. It uses the March Y algorithm. For two steps: initialization (first step) and final-reading-data (last step), address ordering is unimportant. The author uses an LFSR-based address generator during these two steps. It further proposes to modify the RAM structure to use two address decoders – one for regular addressing and another for LFSR-based addresses.

$$March\ Y\ Algorithm\colon \updownarrow (W0);\ \uparrow (R0, W1, R1);\ \downarrow (R1, W0, R0);\ \updownarrow (R0)$$

4.9. Reliability testing

An overvoltage stress test (OVST) can help find latent defects due to imperfections in the insulating layer or weak gate oxide. It is done at the wafer level and helps identify defective parts before reaching an expensive burn-in test [79]. Agrawal *et al.* [37] presented a method to do such OVST testing, which increases supply voltage by 40% during the testing. First, the memories are prefilled with logic 0 or 1 value to avoid contention. Then, scan patterns are applied to test logic. These configurations test memories under static stress and logic under dynamic stress.

Angione *et al.* [38] presented a method to generate a test for burn-in, which targets interconnection to embedded memories. The author developed functional tests to exercise memory as they provide more uniform stress than MBIST-generated tests. For effectiveness, the test needs to have maximum toggle coverage. Instead of checking for all addresses, the two locations where maximum address bits toggling happen are selected. For example, addresses: 0x4001_7FFF → 0x4001_8000 (refer to Figure 11). Address bits are stressed by repetitively performing write and read on these two locations. Analytical evaluation is done using these tests and March C-based MBIST. Execution time for them is 3.14 times better than the March C algorithm.

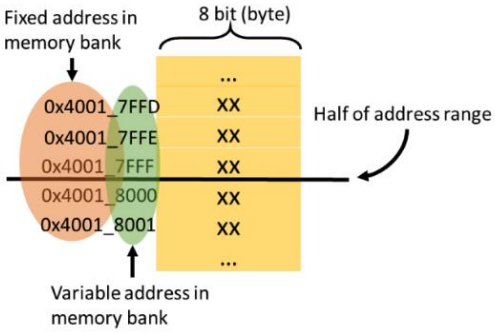

Figure 11. Variable address toggling [38]

SRAMs commonly employ assist circuits like word line underdrive (WLUD) to improve read stability. They are also provided with read-and-write time programmable self-timing controls. The self-timing control can be used to shorten or lengthen read/write window time. Kinseher *et al.* [39] explores using these timing controls to exercise subtle memory delay defects without increasing voltage. It proposes to run MBIST with different configurations of self-timing and assist circuits in screening defects prone to long-term bias temperature instability (BTI) aging.

4.10. FPGA implementation of MBIST

Field programmable gate array (FPGA) implementation is a great choice for chips requiring limited volume. Multiple memory testing architectures were presented to show their usage and efficiency when implemented on FPGA. An MBIST architecture contains a test pattern generator (TPG), a circuit-under-test (CUT) consisting of memories, and an output response analyzer (ORA). The architecture presented in [40] implements a modified March C- algorithm and uses separate TPG and ORA for every CUT. It maps one TPG, CUT, and ORA set onto one configurable logic block (CLB) of FPGA. This mapping eliminates dependency on interconnects. The architecture presented in [80] suffers from such dependency, limiting the tracing of faults in TPG. Further, the use of multiple TPGs eliminates fault aliasing when a single TPG is used and it contains a fault. The architecture improves CLB coverage to 54%, whereas it is 46% for [80].

Another FPGA-based MBIST architecture is presented in [43], which implemented the March 17N algorithm. It shows area efficiency by using only 158 slices of look-up table (LUT) to achieve 497.4 MHz clock speed. Kumari *et al.* [44] presented an FPGA implementation of MBIST. It performs memory testing using the FPGA kit's write and read mode. In addition, it uses an embedded software-based logic analyzer to monitor signals.

5. ALGORITHMIC APPROACHES

Emerging technologies and increasing density give rise to newer defects in memory. Though many algorithms have been presented in the past, detecting the newer defects needs a new algorithm. Multiple algorithms have been published recently, considering the defects as well as improving the efficiency of existing algorithms. This section summarizes them.

5.1. New algorithms

March C algorithm has been the main algorithm being used over the years. Researchers kept optimizing this algorithm to cover newer defects and improve run time. This section presents algorithms that have been presented in recent times.

a) CHECKERMARC [47]: Number of operations per cell = 10
This algorithm combines steps from March C- and Checkerboard algorithms. While the checkerboard algorithm reads 0 and 1 from even and odd locations, March C- performs successive read and write operations. This algorithm writes and reads at even and odd locations. One major advantage of using this algorithm is that it covers transition, coupling, and bridging faults.

b) March Y [49]: Number of operations per cell = 22
The author uses the March C+ algorithm as a baseline and deduces the March Y algorithm through it. Next, it analyzes fault coverage by each step. It then analyzes fault sensitization conditions for WDF, CFdsxwx, and CFWd fault models and shows they can be covered using the proposed algorithm.

c) Modified March 8N [48]: Number of operations per cell = 8
The algorithm is based on the March Y algorithm. It can detect stuck-at, transition, and coupling faults. Additionally, it detects address decoder, retention, and most neighborhood pattern sensitive faults (NPSF).

d) March 5N [52], [50]: Number of operations per cell = 8
The algorithm covers stuck-at, transition, inversion coupling and address decoder faults. It is faster due to the reduced number of operations. Furthermore, the fault coverage achievable using it is higher than the MATS++ algorithm.

e) March SS [51]: Number of operations per cell = 22N
This algorithm is being developed for word-oriented memories. The authors start with a bit-oriented March SS algorithm and modify it for an 8-bit word-oriented algorithm. Applying word-oriented transactions saves huge test time compared to applying bit-oriented transactions. The algorithm covers stuck at, transition, coupling, read disturb, and address decoder faults

f) Modified March Y [53]: Number of operations per cell = 8N
This algorithm is developed with the intent of reducing power during memory testing. Since address generation incurs significant power consumption, the author tried to use the low-power address generation method based on LFSR for the algorithm steps where the order is immaterial. For example, in the March Y algorithm, the first step, W0, and the last step, R0, do not need to perform operations in a particular order of cell addresses. So, the architecture performs these steps using a low-power address generator. Operations in other steps are performed with the required order of addresses.

g) March NS [46]: Number of operations per cell = 10N
This algorithm is a variant of the March algorithm. It has added steps to detect the NPSF faults. Along with NPSF, it detects Struck-at, Transition, Coupling, and Addressing faults.

5.2. Algorithms for pattern sensitive faults testing

Due to increased density, nanometer memories are more prone to faults due to the coupling effects from adjacent cells. Several algorithms are presented to detect such faults. This section summarizes those algorithms. Buslowska and Yarmolik [55] presented a study on detecting pattern-sensitive faults (PSFs) by various algorithms. The fault model detects value corruption in a cell due to particular values in other cells. The author presented a method to find fault coverage by an algorithm for PSF faults involving k cells as influencing cells. For the March test, the maximum fault coverage for different k values is shown in Table 5.

Table 6 shows the fault coverage of PSF for various algorithms. The complexity of the algorithm indicates the number of march commands in it. March PS 18N shows maximum coverage among them, which is 66.6%. Further, MATS++, March C- and March PS 18N are studied for multi-run tests with random

backgrounds. With seven iterations, they achieved coverage of 89.0%, 99.9%, and 99.9 %, respectively. The achieved coverages help conclude that multi-run tests need to be run to increase coverage of PSF fault.

Table 5. Maximum PSF fault coverage for the March test

k	3	5	7	9
FCmax	66.67%	20.00%	5.36%	1.39%

Table 6. Fault coverage of PSF3 [55]

Test	Complexity	FC(PSF3)
MPS (3N)	3N	12.5%
MATS+	4N	12.5%
MPS (5N)	5N	25.0%
MATS++	5N	25.0%
March X	5N	25.0%
March Y	7N	25.0%
March A	14N	33.3%
March B	16N	33.0%
Algorithm B	16N	50.0%
March C-	9N	50.0%
March LA	21N	50.0%
March PS (23N)	22N	66.6%
March PS (18N)	17N	66.6%

Cascaval *et al.* [54] presented an algorithm to detect unlinked static three-cell coupling faults. The first algorithm presented in [81] is long and needs 5.n.log2n+22.5n operations. The paper considers a reduced three-cell coupling model where only cells from physically neighboring regions form a three-cell coupling, as shown in Figure 12. The author presented an algorithm MT-SR3C along with analysis to show coverage of the faults. A read is not required between two consecutive writes if the second write is not transitioned. The algorithm eliminates a few operations for these cases. The algorithm is 18.5% shorter than the March SR3C algorithm.

In paper [57], an analysis of two runs of March tests has been presented for the passive unrestricted neighborhood pattern sensitive faults (PUNPSF) fault model. Algorithm MATS++ and March C- have been run two times. Various address decimations were used for the second run. The study found that maximum fault coverage is obtained when second address sequences with address decimation as two are used. Later Khushi and Singh [58] compared March B and March M algorithms and showed the effectiveness of March M in covering coupling faults.

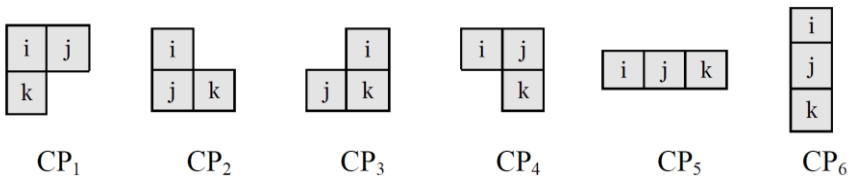

Figure 12. Coupling patterns of three adjacent cells [54]

5.3. MBIST controller with configurable algorithms

Many MBIST architectures implement multiple algorithms to detect modeled and unmodelled faults. However, there needs to be a provision to select a few for usage, like debugging. Implementations in [59], [60], and [34] implement multiple algorithms but provide control to select one for execution

5.4. Algorithms combining

When used together, the March algorithms cover all types of neighborhood cells. However, this is not the case when the algorithms are executed stand-alone. Bui *et al.* [61] presented a combination of March C- and TLAPNPSF algorithms. It showed results that cover all types of faults related to SAF, TF, RDF, IRF, ADF, CFs, active NPSF (ANPSF), passive NPSF (PNPSF), and static NPSF (SNPSF) models using this new algorithm.

6. VALIDATION STRATEGIES

6.1. Collection of diagnostic data

Koshy and Arun [62] presented a method to send out memory built-in self-test failure data for off-chip analysis. It collects and compresses data on a march element basis. Whenever a failure is detected, the data is sent out for diagnosis. The compression reduces the data volume and saves time in shifting out data.

6.2. Algorithm trimming

Wahab *et al.* [63] presented an algorithm trimming methodology for low-cost products with high defects per million (DPM) tolerance levels. Test time optimization starts after a product reaches production-level testing and when sufficient sample size is available. The method uses three algorithms: MarchX, MarchC, and Hammer read for analysis. It removes a few march elements and reruns the test on ATE on sufficient units. The trimming is accepted if the failures are lower than the allowed DPM. It injects errors into memory and cross-checks the trimmed algorithm using simulation. The author achieved a 33% test time reduction using this method.

6.3 Algorithm Verification

The correctness of the operation must be checked whenever a new algorithm is developed. Kinseher *et al.* [5] presented a flow to perform this. First, the algorithm is specified in a higher-level language. Then, the code is simulated using a software method, and memory responses are noted. For verifying the algorithm on a design that implements it, the author suggests simulating to record values on the following signals: i) address bus of the memory, ii) data input and output bus, iii) clock signal, iv) circuit select signal, and v) read or write select signal. Values from circuit simulation and software-based runs must be compared. A parser is used later for this purpose. The author provided detailed steps to use the flow to verify Partial MOVI algorithm implementation.

6.4. Reduced address verification

Multiple MBIST algorithms are implemented in a typical SOC. Few features like bitmapping or repair do not require checking for all memory locations during pre-silicon verification. Bagewadi *et al.* [64] presented a method to speed up simulations without compromising verification quality. The design is modified to make the address range configurable. During testbench generation, a smaller range of addresses is specified. The author performed experiments with a different number of addresses. It showed a 59% improvement in simulation time when 10% of addresses were used.

6.5. Physical failure analysis

Xu *et al.* [65] proposed a defect analysis method. Parts from test escapes from a mass production test were analyzed. First, the flow injected defects by varying resistance values at the bit cell to model open defects. Next, it performed simulations with these circuits. Later, the flow did a physical failure analysis to confirm the observations with simulation results. The defect occurred when the access time of the word line was reduced at high temperature and high voltage conditions during continuous writing of reversed values. The analysis helped to debug the failure and improve memory yield.

7. DISCUSSION

After reviewing the recent advances in the related literature, this section discusses the benefits of various approaches to memory testing. Then, it highlights the main challenges that remain in key aspects. It also discusses a direction for future works.

a. In-field testing: automotive electronic systems have traditionally utilized the well-established technology node for higher yield, reliability, and low cost. The earlier technology maturity period was counted to be about five years on average. However, this margin has significantly reduced in the last decade due to time-to-market pressure. Now, automotive companies have started using a technology that matures in two years. This trend is demanding a higher quality of testing. Chips need to be checked for many defect types to achieve low defective parts per million (DPPM). Test time and cost are becoming challenging due to these checks, and this is demanding more efficient algorithms.
b. New algorithms: a newer semiconductor manufacturing process designs memory with highly dense structures, leading to various defects and causing malfunctions and performance impacts. The dense structures also increase pattern-sensitive faults (PSF). Therefore, precise fault models used to develop efficient test algorithms are crucial. Unfortunately, existing testing algorithms may not detect these defects, and new algorithms must be developed. In addition, user-defined operations are highly desirable to debug any new fault, especially related to smaller geometries. So, MBIST architectures need to have provisions to support such algorithms.

c. Write-through testing: combining conventional memory-BIST and ATPG tend to miss a few paths around memory. This is due to the muxes added to the interfaced collar. The write-through testing techniques reviewed in this paper help to test those paths and improve fault coverage.
d. Test scheduling: applying test algorithms results in a huge amount of toggling around memory. The test scheduler needs to consider the size of memory, clock frequency of operation, and power rating of the package. Also, all other logic and memories, along with controllers, must be prevented from toggling by the disabling clock.
e. Modeling standardization: as new memory technologies keep emerging; their prototyping is required to enable the development of new architectures and systems. IEEE 1450.6.2-2014 standard has been developed, which outlines the format in which details about memory need to be described. Along with interface details, it describes how a write and read operation is performed in the memory. The usage of such models enables consistency across various stages like MBIST insertion, verification, testing, and failure analysis.
f. Validation strategies: design optimization may result in logically adjacent cells being physically apart. The effects of such address and data scrambling during pattern generation and verification need to be considered. Many algorithms are being proposed, and their verification remains a challenge. The usage of multiple data backgrounds and the non-linear traversal of address space need to be considered while developing test patterns.
g. 3D test: the growing trend of building multi-die solutions necessitates finding methodologies to test interconnects between the dies. Popularly, memories are kept on separate dies and are accessed from logic dies. The interconnects consisting of micro-bumps and TSVs are prone to coupling faults. IEEE 1838 defines the structure of wrapper cells to be added on such interconnects. DFT architectures in all these dies need to support testing the interconnects using such wrapper cells. Verification and testing of this infrastructure at the individual die level are crucial to enabling testing defects at the package level.

8. CONCLUSION

Embedded memories play an essential role in today's SoCs. The smaller manufacturing geometries have allowed large memory on the chip. Embedded memories are a critical circuit due to no direct connections to external pins, complicating the test process. This paper presented a survey of various approaches being used for testing them. Built-in self-test is the best solution for testing and diagnosing embedded memories within SoCs. It offers a simple and low-cost means without compromising performance. New defects, fault models, algorithms, in-field, and 3D integrated circuit (IC) testing are reviewed for the principle being used in them. The need and benefits of using them are discussed. This paper is closed with a summary of challenges and future research directions.

REFERENCES

[1] D. Sargsyan, "ISO 26262 compliant memory BIST architecture," in *International Scientific and Technical Conference on Computer Sciences and Information Technologies*, Sep. 2017, pp. 78–82, doi: 10.1109/CSITechnol.2017.8312145.
[2] V. Sontakke and D. Atchina, "Memory built-in self-repair and correction for improving yield: a review," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 14, no. 1, pp. 140–156, Feb. 2024, doi: 10.11591/ijece.v14i1.pp140-156.
[3] G. Tshagharyan, G. Harutyunyan, and Y. Zorian, "An effective functional safety solution for automotive systems-on-chip," in *Proceedings - International Test Conference*, Oct. 2017, pp. 1–10, doi: 10.1109/TEST.2017.8242075.
[4] A. J. Van De Goor, "Using march tests to test SRAMs," *IEEE Design and Test of Computers*, vol. 10, no. 1, pp. 8–14, Mar. 1993, doi: 10.1109/54.199799.
[5] J. Kinseher, M. Richter, and I. Polian, "On the automated verification of user-defined MBIST algorithms," *ZuE 2015: Reliability by Design - 8. GMM/ITG/GI-Fachtagung*, pp. 50–55, 2015.
[6] S. B. Ghale and P. Namita, "Design and implementation of memory BIST for hybrid cache architecture," in *Proceedings of the 6th International Conference on Communication and Electronics Systems*, 2021, pp. 26–31, doi: 10.1109/ICCES51350.2021.9489225.
[7] S. Hamdioui, "Testing embedded memories: a survey," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7721, Springer Berlin Heidelberg, 2013, pp. 32–42, doi: 10.1007/978-3-642-36046-6_4.
[8] G. P. Acharya and M. A. Rani, "Survey of test strategies for system-on chip and it's embedded memories," in *2013 IEEE Recent Advances in Intelligent Computational Systems*, Dec. 2013, pp. 199–204, doi: 10.1109/RAICS.2013.6745473.
[9] G. Harutyunyan and Y. Zorian, "An effective embedded test and diagnosis solution for external memories," in *Proceedings of the 21st IEEE International On-Line Testing Symposium*, Jul. 2015, pp. 168–170, doi: 10.1109/IOLTS.2015.7229852.
[10] I. Jani, D. Lattard, P. Vivet, J. Durupt, S. Thuries, and E. Beigne, "Test solutions for high density 3D-IC interconnects - focus on SRAM-on-logic partitioning," in *Proceedings of the European Test Workshop*, May 2019, vol. 2019-May, doi: 10.1109/ETS.2019.8791531.
[11] D. Sargsyan, "Firmware generation architecture for memory BIST," *2018 IEEE East-West Design & Test Symposium (EWDTS)*, Kazan, Russia, 2018, pp. 1-4, doi: 10.1109/EWDTS.2018.8524853.
[12] A. Becker, "Short burst software transparent on-line MBIST," in *Proceedings of the IEEE VLSI Test Symposium*, Apr. 2016, pp. 1–6, doi: 10.1109/VTS.2016.7477287.

[13] S. Im, G. Nam, S. Park, and M. Noh, "Advanced safety test solution for automotive SoC based on in-system-test architecture," in *Proceedings - IEEE International Symposium on Circuits and Systems*, May 2022, pp. 2290–2293, doi: 10.1109/ISCAS48785.2022.9937235.
[14] F. Angione *et al.*, "Online scheduling of concurrent memory BISTs execution at real-time operating-system level," in *Proceedings - IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, DFT*, Oct. 2022, vol. 2022-Octob, doi: 10.1109/DFT56152.2022.9962338.
[15] P. K. Datla Jagannadha *et al.*, "Special session: in-system-test (IST) architecture for NVIDIA Drive-AGX platforms," *2019 IEEE 37th VLSI Test Symposium (VTS)*, Monterey, CA, USA, 2019, pp. 1-8, doi: 10.1109/VTS.2019.8758636.
[16] N. Mishra, N. Naresh, and A. Acharya, "Parallel field test architecture for boot-ROMs in safety-critical SoCs," *2021 IEEE International Test Conference India (ITC India)*, Bangalore, India, 2021, pp. 1-6, doi: 10.1109/ITCIndia52672.2021.9532633.
[17] B. Bhaskaran, S. Chadalavada, S. Sarangi, N. Valentine, V. A. R. Nerallapally, and A. Abdollahian, "At-speed capture global noise reduction and low-power memory test architecture," Apr. 2017, doi: 10.1109/VTS.2017.7928936.
[18] K. S. Das and P. Prakash, "Automatic MBIST scheduling engine," *2019 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, Bangalore, India, 2019, pp. 1-6, Jul. 2019, doi: 10.1109/CONECCT47791.2019.9012926.
[19] R. Zeli, R. Silveira, and Q. Qureshi, "SoC Memory test optimization using NXP MTR solutions," *2019 IEEE Latin American Test Symposium (LATS)*, Santiago, Chile, 2019, pp. 1-5, doi: 10.1109/LATW.2019.8704566.
[20] C. Hu, X. Li, Z. Fu, Q. Tang, and R. Zhao, "The implementation of a configurable MBIST controller for multi-core SoC," in *Communications in Computer and Information Science*, vol. 1146, Springer Singapore, 2019, doi: 10.1007/978-981-15-1850-8_8.
[21] R. Nourmandi-Pour, "A programmable IEEE 1500-compliant wrapper for testing of word-oriented memory cores," *Journal of Circuits, Systems and Computers*, vol. 27, no. 9, Apr. 2018, doi: 10.1142/S0218126618501347.
[22] P. Arora, P. Gallagher, and S. L. Gregor, "Core test language based high quality memory testing and repair methodology," in *2021 IEEE International Test Conference India (ITC India)*, Jul. 2021, pp. 1–6, doi: 10.1109/ITCIndia52672.2021.9532722.
[23] K. R. Raval, Y. D. Parmar, and C. R. Panchal, "DFT methodology for memory testing on lower technological node," in *Proceedings of the International Conference on Computing Methodologies and Communication*, Jul. 2017, pp. 430–435, doi: 10.1109/ICCMC.2017.8282725.
[24] B. Mohammad, "Embedded memory interface logic and interconnect testing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 9, pp. 1946–1950, Sep. 2015, doi: 10.1109/TVLSI.2014.2354381.
[25] Y. Gao, T. Zhang, P. Pokharel, S. Chakraborty, and D. M. H. Walker, "Pseudo functional path delay test through embedded memories," *Journal of Electronic Testing: Theory and Applications (JETTA)*, vol. 31, no. 1, pp. 35–42, Dec. 2015, doi: 10.1007/s10836-014-5497-x.
[26] J. Park, J. H. Lee, S. K. Park, K. C. Chun, K. Sohn, and S. Kang, "An in-DRAM BIST for 16 Gb DDR4 DRAM in the 2nd 10-nm-Class DRAM process," *IEEE Access*, vol. 9, pp. 33487–33497, 2021, doi: 10.1109/ACCESS.2021.3061349.
[27] V. Sridharan *et al.*, "Memory errors in modern systems," *ACM SIGPLAN Notices*, vol. 50, no. 4, pp. 297–310, Mar. 2015, doi: 10.1145/2775054.2694348.
[28] H. Y. Yang *et al.*, "Testing methods for quaternary content addressable memory using charge-sharing sensing scheme," *2015 IEEE International Test Conference (ITC)*, Anaheim, CA, USA, 2015, pp. 1-10, doi: 10.1109/TEST.2015.7342409.
[29] O. S. Nisha and K. Sivasankar, "Architecture for an efficient MBIST using modified March-y algorithms to achieve optimized communication delay and computational speed," *International Journal of Pervasive Computing and Communications*, vol. 17, no. 1, pp. 135–147, Jan. 2021, doi: 10.1108/IJPCC-05-2020-0032.
[30] S. Vennelakanti and S. Saravanan, "Design and analysis of low power memory built in self test architecture for SoC based design," *Indian Journal of Science and Technology*, vol. 8, no. 14, Jul. 2015, doi: 10.17485/ijst/2015/v8i14/62716.
[31] S. Saravanan, M. Hailu, G. M. Gouse, M. Lavanya, and R. Vijaysai, "Design and analysis of low-transition address generator," in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, vol. 274, Springer International Publishing, 2019, pp. 239–247, doi: 10.1007/978-3-030-15357-1_19.
[32] V. Suresh Kumar and R. Manimegalai, "Efficient memory built in self test address generator implementation," *International Journal of Applied Engineering Research*, vol. 10, no. 7, pp. 16797–16813, 2015.
[33] S. S. R. Dannina and U. V. R. Kumari, "Error tolerant MBIST design with efficient high throughput and low power reversible techniques," in *Proceedings - 2021 IEEE 10th International Conference on Communication Systems and Network Technologies*, Jun. 2021, pp. 230–235, doi: 10.1109/CSNT51715.2021.9509611.
[34] L. Zhang, Z. Wang, Y. Li, and L. Mao, "A precise design for testing high-speed embedded memory using a BIST circuit," *IETE Journal of Research*, vol. 63, no. 4, pp. 473–481, Feb. 2017, doi: 10.1080/03772063.2017.1285259.
[35] P. Ramakrishna, T. Vamshika, and M. Swathi, "FPGA implementation of memory bists using single interface," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 9, no. 3, pp. 55–58, Sep. 2020, doi: 10.35940/ijrte.b3975.099320.
[36] T. McLaurin and R. Knoth, "The challenges of implementing an MBIST interface: A practical application," *2019 IEEE International Test Conference (ITC)*, Washington, DC, USA, 2019, pp. 1-6, doi: 10.1109/ITC44170.2019.9000157.
[37] S. Agrawal, K. Pandey, R. C. Kumar Y, and A. Jain, "An efficient test architecture for concurrent over voltage stress testing (OVST) of logic and memory," Jul. 2021, doi: 10.1109/ITCIndia52672.2021.9532907.
[38] F. Angione *et al.*, "An optimized burn-in stress flow targeting interconnections logic to embedded memories in automotive systems-on-chip," May 2022, doi: 10.1109/ETS54262.2022.9810396.
[39] J. Kinseher, M. Voelker, and I. Polian, "Improving testability and reliability of advanced SRAM architectures," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 3, pp. 456–467, Jul. 2019, doi: 10.1109/TETC.2017.2677400.
[40] P. A. Babu and J. Samudrala, "An advanced BIST architecture with low power LBIST and BDS oriented march algorithm for intra word coupling faults," *International Journal of Engineering and Computer Science*, vol. 4, no. 1, pp. 9809–9813, 2015.
[41] P. K. John and R. Antony, "BIST architecture for multiple RAMs in SoC," *Procedia Computer Science*, vol. 115, pp. 159–165, 2017, doi: 10.1016/j.procs.2017.09.121.
[42] M. Sasikumar, R. Bhakthavatchalu, K. N. Sreehari, and A. S. Kumar, "Scalable and rapid fault detection of memories using MBIST and signature analysis," in *Lecture Notes in Electrical Engineering*, vol. 703, Springer Singapore, 2021, pp. 351–367, doi: 10.1007/978-981-15-8391-9_26.
[43] M. A. Ahmed, D. Elizabeth Rani, and S. A. Sattar, "FPGA based high speed memory BIST controller for embedded applications," *Indian Journal of Science and Technology*, vol. 8, no. 33, Dec. 2015, doi: 10.17485/ijst/2015/v8i33/76080.
[44] K. L. V Ramana Kumari, M. Asha Rani, and N. Balaji, "FPGA implementation of memory design and testing," in *Proceedings - 7th IEEE International Advanced Computing Conference*, Jan. 2017, pp. 552–555, doi: 10.1109/IACC.2017.0119.
[45] R. Manasa, R. Verma, and D. Koppad, "Implementation of BIST Technology using March-LR Algorithm," in *2019 4th IEEE*

International Conference on Recent Trends on Electronics, Information, Communication and Technology, May 2019, pp. 1208–1212, doi: 10.1109/RTEICT46194.2019.9016784.
[46] G. S. Lakshmi, N. K, and C. Subhas, "A march ns algorithm for detecting all types of single bit errors in memories," *Journal of Emerging Technologies and Innovative Research*, vol. 6, no. 4, pp. 289–296, 2019.
[47] A. Pundir and O. P. Sharma, "CHECKERMARC: a modified novel memory-testing approach for bit-oriented SRAM," *International Journal of Applied Engineering Research*, vol. 12, pp. 3023–3028, 2017.
[48] M. Vishnoi, A. Kumar, and M. Sanadhya, "Design of improved built-in-self-test algorithm (8n) for single port memory," *International Journal of Soft Computing and Engineering (IJSCE)*, pp. 2231–2307, 2012.
[49] Y. Wang, Q. Zheng, and Y. Yuan, "The improvement of march C+ algorithm for embedded memory test," in *Communications in Computer and Information Science*, vol. 592, Springer Berlin Heidelberg, 2016, pp. 31–37, doi: 10.1007/978-3-662-49283-3_4.
[50] K. H. Ng, N. E. Alias, A. Hamzah, M. L. P. Tan, U. U. Sheikh, and Y. A. Wahab, "A March 5n FSM-based memory built-In Self-test (MBIST) architecture with diagnosis capabilities," in *IEEE International Conference on Semiconductor Electronics, Proceedings, ICSE*, Aug. 2022, pp. 69–72, doi: 10.1109/ICSE56004.2022.9863160.
[51] G. P. Acharya, M. A. Rani, G. G. Kumar, and L. Poluboyina, "Adaptation of March-SS algorithm to word-oriented memory built-in self-test and repair," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 26, no. 1, pp. 96–104, Apr. 2022, doi: 10.11591/ijeecs.v26.i1.pp96-104.
[52] T. S. Nguan Kong *et al.*, "An efficient march (5n) FSM-based memory built-in self test (MBIST) architecture," in *Proceedings - 2021 IEEE Regional Symposium on Micro and Nanoelectronics, RSM 2021*, Aug. 2021, pp. 76–79, doi: 10.1109/RSM52397.2021.9511602.
[53] O. S. Nisha and K. Sankar, "VLSI architecture for an efficient memory built in self test for configurable embedded SRAM memory," *International Journal of Control Theory and Applications*, vol. 9, pp. 367–380, 2016.
[54] P. Caşcaval and D. Caşcaval, "March test algorithm for unlinked static reduced three-cell coupling faults in random-access memories," *Microelectronics Journal*, vol. 93, Nov. 2019, doi: 10.1016/j.mejo.2019.104619.
[55] E. Buslowska and V. Yarmolik, "Multi-run march tests for pattern sensitive faults in RAM," *2018 IEEE East-West Design & Test Symposium (EWDTS)*, Kazan, Russia, 2018, pp. 1-6, doi: 10.1109/EWDTS.2018.8524643.
[56] I. Mrozek and V. Yarmolik, "Pseudo-exhaustive random access memory testing based on march tests with random background variation," *2018 IEEE East-West Design & Test Symposium (EWDTS)*, Kazan, Russia, 2018, pp. 1-8, doi: 10.1109/EWDTS.2018.8524824.
[57] I. Mrozek and V. Yarmolik, "Two-run RAM march testing with address decimation," *Journal of Circuits, Systems and Computers*, vol. 26, no. 2, Art. no. 1750031, Nov. 2017, doi: 10.1142/S0218126617500311.
[58] Khushi and K. Singh, "Performance analysis of march M and B algorithms for memory built-in self-test (BIST)," in *2022 IEEE World Conference on Applied Intelligence and Computing (AIC)*, Jun. 2022, pp. 78–84, doi: 10.1109/AIC55036.2022.9848869.
[59] A. A. Wojciechowski, K. Marcinek, and W. A. Pleskacz, "Configurable MBIST processor for embedded memories testing," in *2019 MIXDES - 26th International Conference "Mixed Design of Integrated Circuits and Systems,"* Jun. 2019, pp. 341–344, doi: 10.23919/MIXDES.2019.8787161.
[60] N. Maneshinde, P. Hegade, R. Mittal, N. Palecha, and M. S. Suma, "Programmable FSM based built-in-self-test for memory," in *2016 IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology (RTEICT)*, May 2016, pp. 194–199, doi: 10.1109/RTEICT.2016.7807811.
[61] T. Q. Bui, L. D. Pham, H. M. Nguyen, V. T. Nguyen, T. C. Le, and T. Hoang, "An effective architecture of memory built-in self-test for wide range of SRAM," in *2016 International Conference on Advanced Computing and Applications (ACOMP)*, Nov. 2016, pp. 121–124, doi: 10.1109/ACOMP.2016.026.
[62] T. Koshy and C. S. Arun, "Diagnostic data detection of faults in RAM using different march algorithms with BIST scheme," in *2016 International Conference on Emerging Technological Trends (ICETT)*, Oct. 2016, pp. 1–6, doi: 10.1109/ICETT.2016.7873754.
[63] A. A. A. Wahab, S. S. N. Alhady, W. A. F. W. Othman, H. Husin, and N. Q. M. Adnan, "Optimizing RAM testing method for test time saving using automatic test equipment," in *Lecture Notes in Electrical Engineering*, vol. 829 LNEE, Springer Singapore, 2022, pp. 260–266, doi: 10.1007/978-981-16-8129-5_41.
[64] S. N. Bagewadi, S. Shadab, and J. Roopa, "Fast BIST mechanism for faster validation of memory array," in *2019 4th IEEE International Conference on Recent Trends on Electronics, Information, Communication and Technology*, May 2019, pp. 61–65, doi: 10.1109/RTEICT46194.2019.9016882.
[65] T. Xu, J. Huang, M. Bu, and Z. Jiang, "An SRAM test quality improvement method for automotive chips," *2021 IEEE International Test Conference in Asia (ITC-Asia)*, Shanghai, China, 2021, pp. 1-4, doi: 10.1109/ITC-Asia53059.2021.9808542.
[66] L. Jiang, Y. Liu, L. Duan, Y. Xie, and Q. Xu, "Modeling TSV open defects in 3D-stacked DRAM," *2010 IEEE International Test Conference*, Austin, TX, USA, 2010, pp. 1-9, doi: 10.1109/TEST.2010.5699217.
[67] R. Wang and K. Chakrabarty, "Testing of interposer-based 2.5D integrated circuits," *2016 IEEE International Test Conference (ITC)*, Fort Worth, TX, USA, 2016, pp. 1-10, doi: 10.1109/TEST.2016.7805875.
[68] B. Nadeau-Dostie, A. Sitburt, and V. K. Agarwal, "A serial interfacing technique for built-in and external testing of embedded memories," *1989 Proceedings of the IEEE Custom Integrated Circuits Conference*, San Diego, CA, USA, 1989, pp. 22.2/1-22.2/5, doi: 10.1109/CICC.1989.56808.
[69] W. B. Jone, D. C. Huang, S. C. Wu, and K. J. Lee, "An efficient BIST method for distributed small buffers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 10, no. 4, pp. 512–514, Aug. 2002, doi: 10.1109/TVLSI.2002.800532.
[70] A. A. Hwang, I. A. Stefanovici, and B. Schroeder, "Cosmic rays don't strike twice," *ACM SIGARCH Computer Architecture News*, vol. 40, no. 1, pp. 111–122, Mar. 2012, doi: 10.1145/2189750.2150989.
[71] H. H. Wu, J. N. Lee, M. C. Chiang, P. W. Liu, and C. F. Wu, "A comprehensive TCAM test scheme: an optimized test algorithm considering physical layout and combining scan test with at-speed BIST design," *2009 International Test Conference*, Austin, TX, USA, 2009, pp. 1-10, doi: 10.1109/TEST.2009.5355536.
[72] S. Barbagallo, A. Burri, D. Medina, P. Camurati, P. Prinetto, and M. Sonza Reorda, "An experimental comparison of different approaches to ROM BIST," in *Proceedings, Advanced Computer Technology, Reliable Systems and Applications*, 1991, pp. 567–571, doi: 10.1109/cmpeur.1991.257450.
[73] D. Park, T. G. Kim, G. Cho, K. Lee, and C. Kim, "A safe microcontroller with silent CRC calculation hardware for code ROM integrity verification in IEC-60730 class-B," in *The 1st IEEE Global Conference on Consumer Electronics 2012*, Oct. 2012, pp. 197–200, doi: 10.1109/GCCE.2012.6379577.
[74] V. Sontakke and J. Dickhoff, "Developments in scan shift power reduction: a survey," *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 12, no. 6, pp. 3402–3415, Dec. 2023, doi: 10.11591/eei.v12i6.5668.

[75] V. Sontakke and J. Dickhoff, "A survey of scan-capture power reduction techniques," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13, no. 6, pp. 6118–6130, Dec. 2023, doi: 10.11591/ijece.v13i6.pp6118-6130.
[76] K. Jamal, K. M. Chari, and P. Srihari, "Test pattern generation using thermometer code counter in TPC technique for BIST implementation," *Microprocessors and Microsystems*, vol. 71, Nov. 2019, doi: 10.1016/j.micpro.2019.102890.
[77] K. Jamal and P. Srihari, "Low power TPC using BSLFSR," *International Journal of Engineering and Technology*, vol. 8, no. 2, pp. 759–767, 2016.
[78] K. Jamal and P. Srihari, "Analysis of test sequence generators for built-in self-test implementation," in *2015 International Conference on Advanced Computing and Communication Systems*, Jan. 2015, pp. 1–4, doi: 10.1109/ICACCS.2015.7324096.
[79] M. P. L. Ooi, Z. A. Kassim, and S. N. Demidenko, "Shortening burn-in test: Application of HVST and weibull statistical analysis," *IEEE Transactions on Instrumentation and Measurement*, vol. 56, no. 3, pp. 990–999, Jun. 2007, doi: 10.1109/TIM.2007.894165.
[80] B. Dutton and C. Stroud, "Built-in self-test of embedded seu detection cores in virtex-4 and virtex-5 FPGAs," in *Proceedings of the 2009 International Conference on Embedded Systems and Applications*, 2009, pp. 149–155.
[81] R. Nair, S. M. Thatte, and J. A. Abraham, "Efficient algorithms for testing semiconductor random-access memories," *IEEE Transactions on Computers*, vol. C–27, no. 6, pp. 572–576, Jun. 1978, doi: 10.1109/TC.1978.1675150.

BIOGRAPHIES OF AUTHORS

Vijay Sontakke received the BE degree in electronics engineering from Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded, India, in 1997. He received the M.Tech. degree in electronics engineering from Visvesvaraya National Institute of Technology, Nagpur, India, in 1999. He is currently a design engineer at Intel Corporation, Allentown, PA, USA. Before Intel, he worked at Motorola, AMD, Conexant, and Ikanos Communications. He possesses VLSI design experience focusing on the design-for-test. He worked for the entire life cycle of the design-for-test, from test specification definition to production test program release, defined architecture for several SoCs, and led implementation. His current research interests include test power minimization, pattern count optimization, JTAG/IJTAG, 3D IC testing, memory testing, and scan architecture. He can be contacted at email: vijay.sontakke@intel.com.

Delsikreo Atchina received a bachelor of science in engineering from Messiah University, Grantham, USA, in 1998. He received the master of science in electrical engineering from Temple University, Philadelphia, USA, in 2003. He has been a senior DFT engineer at Intel Networking ASIC Group, Allentown, PA, USA, since 2014. He works for multimillion gate blocks and SOCs to integrate test structures to support scan, IJTAG, and memory testing. He also works for validation involving pattern generation, simulation, and test coverage analysis. Before Intel, he worked at lucent, Agere, and LSI Corporation. Delsi also has extensive experience in timing closure and in place and route to address signal integrity issues and physical design rules for large SOCs. He can be contacted at email: delsi.atchina@intel.com.