Reduced March SR Algorithm for Deep-Submicron SRAM Testing

Aiman Zakwan Jidin  
Faculty of Electronics Engineering Technology  
Universiti Malaysia Perlis  
Arau, Malaysia  
aimanzakwan@student.unimap.edu.my

Razaidi Hussin  
Faculty of Electronics Engineering Technology  
Universiti Malaysia Perlis  
Arau, Malaysia  
shidee@unimap.edu.my

Mohd Syafiq Mispan  
Fakulti Teknologi Kejuruteraan Elektrik dan Elektronik  
Universiti Teknikal Malaysia Melaka  
Melaka, Malaysia  
syafiq.mispan@utm.edu.my

Lee Weng Fook  
Emerald System Design Center  
Penang, Malaysia  
seanlee@emersysdesign.com

Loh Wan Ying  
Faculty of Electronics Engineering Technology  
Universiti Malaysia Perlis  
Arau, Malaysia  
wyloh@student.unimap.edu.my

Abstract—Memory Built-In Self-Test (BIST) is a common method to test embedded memories on a chip owing to its fast and low-cost testing. Its efficiency depends on the complexity and the fault coverage offered by its implemented test algorithm. Test algorithms with higher complexities result in longer test times and higher costs. Hence, many research works were proposed to reduce the complexity of the existing algorithm while preserving its fault coverage. This paper presents a method to reduce the March SR algorithm complexity (14N) by analyzing and removing redundant test operations from its test sequence. It was targeted for fault detection in deep-submicron SRAM. The newly produced March mSR algorithm, with 1N complexity lesser than the March SR algorithm but offers the same coverage of the intended faults, was implemented in a Memory BIST controller. The simulations successfully validated its test time reduction, functionality, and fault coverage.

Keywords—Design for Testability, SRAM, Memory BIST, March algorithm, Fault Detection

I. INTRODUCTION

Embedded memories have become the essential component in a System-on-Chip (SoC) nowadays, where they could take up more than 90% of the overall chip area [1], [2]. So, the chip’s quality largely depends on the memory’s yield [3]. With the advent of Very Deep-Submicron (VDSM) technology for chip manufacturing, today’s memory test has become more challenging since more defects may occur in the manufactured memories [4]. Besides, the test length is also a significant concern in the industry since it directly contributes to the test cost and the chip production cost [5], [6].

Memory BIST (MBIST) is commonly used in the industry for embedded memory testing. It is the on-chip circuitry that can self-generate the test and self-check the test responses. It provides a faster test than an external tester like the Automatic Test Generator (ATE). In addition, the test cost is reduced since the dependency on a high-performance and expensive ATE is diminished [1]. The efficiency of an MBIST depends on the test time and coverage. They are determined by the applied test algorithm, which defines the sequence of operations performed on every memory cell.

March algorithms are common choices in the industry owing to their linear complexities and offer better fault coverages than the conventional test algorithms like WALKING I/O and Checkerboard [7]. Its complexity is in the order of N, the memory size under test in Bytes. A March algorithm defines a series of reading (r) and writing (w) operations performed on each memory cell, either in the ascending memory address order (⇑) or in the descending address order (⇓). In the case of a simple or Single-Cell Fault (SCF) detection, the address order is not significant (⇓).

Many previous works had minimized the complexity of the existing March algorithms. The March C algorithm’s complexity was reduced from 11N to 10N by removing an unnecessary read operation to become the March C- algorithm [8]. Its complexity was further reduced to 8N in [9] by having two subgroups that can be executed in parallel. The research in [10] proposed the parallel execution of the March Y algorithm’s second and third elements to reduce its complexity by 3N. Yet, these algorithms were inadequate for detecting newer faults introduced by the VDSM technologies, which are more relevant to today’s memories [4].

This paper presents the March SR algorithm complexity reduction from 14N to 13N, which provides the same fault coverage within a shorter test time. It was achieved by identifying removable redundant test operations within the test sequence through a fault detection analysis on the March SR algorithm. The newly minimized March SR algorithm, referred to as March mSR, was produced as the output. This research focuses on detecting six SCFs: Stuck-At Fault (SAF), Transition Fault (TF), Write Disturb Fault (WDF), and Read Destructive Fault (RDF), Incorrect Read Fault (IRF), Transition Fault (TF), Write Disturb Fault (WDF), and Deceptive Read Destructive Fault (DRDF). Three Double-Cell Faults (DCF) are also covered: Transition Coupling Fault (CFtr), Write Destructive Coupling Fault (CFwd), and Deceptive Read Destructive Coupling Fault (CFdrd).

The description of the March SR algorithm is presented in Section II. Section III discusses the fault detection analysis to detect redundant test operations within the test sequence and the implementation of the newly generated March mSR in the MBIST controller. Section IV presents the validation of its functionality, test completion time, and fault coverage via simulations. The MBIST controller was implemented using Mentor Graphics Tessent MemoryBIST software, while its simulations were performed using the QuestaSim simulator.
TABLE I.  
MARCH SR ALGORITHM DESCRIPTION

<table>
<thead>
<tr>
<th>Test Element</th>
<th>Address Order</th>
<th>Test Operations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0</td>
<td>$w_0$</td>
<td>$r_0, w_0, r_1$</td>
<td>Each cell’s content is set to 0.</td>
</tr>
<tr>
<td>E1</td>
<td>$r_0, w_1, r_1,$</td>
<td>$w_0$</td>
<td>Each cell is read (expecting 0), overwritten to 1, reread (expecting 1), and rewritten to 0, starting from the first cell.</td>
</tr>
<tr>
<td>E2</td>
<td>$r_0, r_0$</td>
<td></td>
<td>Each cell is read twice, starting from the first cell.</td>
</tr>
<tr>
<td>E3</td>
<td>$w_1$</td>
<td></td>
<td>Each cell’s content is set to 1 starting from the first cell.</td>
</tr>
<tr>
<td>E4</td>
<td>$r_1, w_0, r_0,$</td>
<td>$w_1$</td>
<td>Each cell is read (expecting 1), overwritten to 0, reread (expecting 0), and rewritten to 1, starting from the first cell.</td>
</tr>
<tr>
<td>E5</td>
<td>$r_1, r_1$</td>
<td></td>
<td>Each cell is read twice, starting from the last cell.</td>
</tr>
</tbody>
</table>

II. MARCH SR ALGORITHM

Table I describes the March SR algorithm’s test sequence. It has 6 test elements, separated from each other by a semicolon and notated as Eᵢ, where $i = \{0, 1, 2, 3, 4, 5\}$. Table II shows the March SR fault coverage. It has a full coverage of SAF, RDF, IRF, TF, DRDF, and CFtr, and 50% coverage of CFdrd. However, it cannot detect WDF and CFwd since its test sequence does not contain any non-transition write operation. Each SCF has two Fault Primitives (FP), which can be written as $<S/F/R>$ where $S$ is the sensitizing operation, $F$ is the state of the faulty cell, and $R$ is the output value. Thus, two detections are needed for each SCF. In contrast, each DCF, which involves two different cells named the aggressor ($a$) and the victim cells ($v$), has 4 FPs since its sensitization depends on $a$ and $v$ states or operations (notated as $S_a, S_v$). An aggressor can be located either before ($a<v$) or after the victim cell ($a>v$) inside the memory. So, there are eight detections for each DCF since.

III. RESEARCH METHODOLOGY

Fig. 1 illustrates the overall research flow to minimize the March SR algorithm’s complexity. Firstly, its test sequence, which was stored in an input text file, was read by a fault analyzer to conduct the fault detection analysis [11]. Next, further examination was done to identify any unutilized or redundant test operation(s) for the intended fault detection. When a redundant test operation was identified, it was removed from the test sequence, and the newly reduced March SR algorithm was reanalyzed to ensure that it has identical fault coverage to the existing March SR algorithm. Otherwise, no further action was taken. Finally, the current and the reduced March SR algorithms were implemented in the MBIST controller and validated via simulations.

A. March SR Fault Detection Analysis

The March SR algorithm fault detection analysis was executed by the fault analyzer, which was developed in [11], by identifying all possible sensitizer and detector pairs for each fault within the analyzed test sequence. This identification was performed based on the required test operations to detect each fault model [11]–[13], as summarized in Table III. $X$ and $X'$ represent any operations that produce $x$ and $x'$ logics, respectively, as the memory cells’ contents.

In this analysis, the location of the identified fault sensitizer and detector are notated as $E_{i,j}$, which indicates the $j$th test operation in the $E_i$ test element. Table IV shows the March SR algorithm’s fault detection analysis results. The analysis on SAF, RDF, and IRF was grouped since they have the same required test operations for detection. In addition, the analyses on WDF and CFwd were not shown since they are undetectable by the March SR algorithm.
A test operation is removable from the test sequence when all of the following conditions are satisfied:

- **Condition 1** – Each FP for each detectable fault must have at least two pairs of sensitizer and detector.
- **Condition 2** – A test operation is removable if it is unusable for other faults’ detections.
- **Condition 3** – The removal of a test operation must preserve the fault coverage.

Multiple pairs of sensitizer and detector were observed for each FP of SAF and TF, and thus, satisfying Condition 1. Except for the $E_{11}$ and $E_{13}$, the other test operations that can sensitize and detect SAF and TF are also necessary for CFtr detection. Hence, they cannot be removed from the test sequence since Condition 2 was not satisfied. Only the test operations at $E_{01}$ and $E_{11}$ fulfill both Condition 1 and Condition 2. However, the memory cells need to be initialized to 0s through the $E_{01}$ operation (w0) to sensitize and detect CF tr $<0w1/0/>$ and CFtr$<0w1/0/>$ at $E_{12}$ and $E_{13}$. So, it cannot be removed since Condition 3 was not satisfied.

Therefore, only the $E_{11}$ operation (r0) is removable from the test sequence. By eliminating it, the test sequence of the newly reduced March SR algorithm, known as the March mSR, becomes $f(w0)$; $f(w1, r1, w0)$; $f(r0, r0)$; $f(r1, w0)$; $f(r1, r1)$, with 13N of complexity.

**B. March SR and March mSR Implementation in MBIST Controller**

Next, the March SR and March mSR were implemented as the User-Defined Algorithm (UDA) in separated MBIST controllers. Their insertion process was done using the Mentor Graphic Tessent MemoryBIST software. Firstly, the UDA description file for each algorithm was created in a format recognizable by the Tessent software to define the test setup and the test operation sequence at each test element. Next, the Tessent software read the UDA description file during the MBIST insertion process to hard-code the intended test sequence in the MBIST controller. The insertion was configured for a 1 kB Single-Port SRAM as the memory to be tested. In addition, upon completion of the MBIST controller insertion, the Tessent software also generated the test benches and patterns that can be used later on during the simulations.

**IV. RESULTS AND DISCUSSION**

The simulations were conducted on both implemented MBIST controllers using the QuestaSim simulator. In these simulations, a 1 kB Single-Port SRAM was used as the memory model. Therefore, $N$ equals 1024 and the clock period $T_{clk}$ used during the simulations is 20 ns.

Fig. 2 and Fig. 3 show the waveforms from the simulations performed on the MBIST controller with the March mSR and the March SR, respectively, as the UDAs. The output read value dout was compared to the expected value BIST_EXPECT_DATA at every clk cycle whenever the CMP_EN flag was high. The ERROR flag stayed low during both simulations, indicating no mismatch between the observed and expected values had occurred. In addition, a 266240 ns of test completion time was measured between the first and the final test operations of the implemented March mSR algorithm. Subsequently, its expected 13N complexity was derived using (1). While Fig. 3 shows the test using the March SR required 286720 ns for completion. Hence, it demonstrates that a test on a 1 kB memory using the March mSR algorithm required 1024 or $N$ clock cycles less than the test using the March SR algorithm. This difference could be more significant for a larger quantity of memory to be tested with higher storage capacity.

$$\text{Test Completion time} = \text{Complexity} \times N \times T_{clk} \quad (1)$$

Next, the fault detection simulation on the MBIST controller with the March mSR as the UDA was performed to validate its fault coverage. It used a faulty 1 kB Single-Port SRAM as the memory under test, where the memory model previously used in the first simulation was modified to replicate the fault occurrences at the simulation level. The addresses of the victim and the aggressor cells for each FP were chosen arbitrarily. 36 flags were also created to represent the detection of each FP for all faults.

Fig. 4 displays the waveform obtained from the executed fault detection simulation. The detection flags’ values were observed at the end of the simulation, and each fault’s coverage was derived by counting the number of high bits in its fault detection flag. It has 100% coverage of SAF, TF, RDF, DRDF, and CFtr, 50% coverage of CFdrd, and no coverage of WDF and CFwd, which are the same as its expected fault coverage presented in Table II. Table V shows that, with a lesser 1N complexity, the proposed March mSR

<table>
<thead>
<tr>
<th>Fault / RFD / IRF</th>
<th>TF</th>
<th>DRDF</th>
<th>CFtr</th>
<th>CFdrd</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;0/1/&gt; or &lt;0/1/&gt;</td>
<td>NA</td>
<td>(E_{11}, E_{13})</td>
<td>(E_{12}, E_{13})</td>
<td>(E_{12}, E_{13})</td>
</tr>
<tr>
<td>&lt;1/0/&gt; or &lt;1/0/&gt;</td>
<td>NA</td>
<td>(E_{11}, E_{13})</td>
<td>(E_{12}, E_{13})</td>
<td>(E_{12}, E_{13})</td>
</tr>
<tr>
<td>&lt;0w1/0/&gt;</td>
<td>NA</td>
<td>(E_{11}, E_{13})</td>
<td>(E_{12}, E_{13})</td>
<td>(E_{12}, E_{13})</td>
</tr>
<tr>
<td>&lt;1/0/&gt;</td>
<td>NA</td>
<td>(E_{11}, E_{13})</td>
<td>(E_{12}, E_{13})</td>
<td>(E_{12}, E_{13})</td>
</tr>
</tbody>
</table>

**TABLE IV. MARCH SR FAULT DETECTION ANALYSIS**

Fig. 2. Memory BIST simulation waveform (March mSR as UDA)

Fig. 3. Memory BIST simulation waveform (March SR as UDA)
has better DRDF and CFdrd coverages than the March LR [14] but has poorer CFdrd coverage than the March C+ [15].

Therefore, the outputs from these simulations proved that the proposed March mSR was successfully implemented in the MBIST controller. It functions correctly and provides the same fault coverage at a shorter test time than the March SR algorithm. Still, several enhancements are required to improve its detection of the intended faults. Its Data Background (DB) and test operation sequences can be optimized, as proposed in [16], to enhance its fault coverage while preserving its complexity. Besides, the proposed method can also be deployed on other existing March algorithms like March LR (14N) and March C+ (14N), which may produce a new algorithm with lower complexity and better fault coverage.

V. CONCLUSION

This paper presented the generation of the March mSR, a reduced March SR algorithm. The complexity reduction from 14N to 13N was achieved by identifying and removing a redundant test operation in the March SR test sequence through a fault detection analysis, which identifies all possible test operations to sensitize and detect all targeted faults. The newly produced March mSR algorithm was successfully implemented in an MBIST controller. Its functionality and fault coverage were validated via simulations, which also proved that the test with March mSR as the UDA produces an N-clock-cycle shorter test completion time than the March SR algorithm yet provides the same fault coverage.

ACKNOWLEDGMENT

The authors wish to acknowledge the support from Universiti Malaysia Perlis, Universiti Teknikal Malaysia Melaka, and the Malaysian Ministry of Higher Education for this research.

REFERENCES


