Adaptation of March-SS algorithm to word-oriented memory built-in self-test and repair

Gobinda Prasad Acharya[1], Muddapu Asha Rani[2], Ganjikunta Ganesh Kumar[1], Lavanya Poluboyina[1]
[1]Department of Electronics and Communication Engineering, Sreenidhi Institute of Science and Technology, Hyderabad, India
[2]Department of Electronics and Communication Engineering, JNTUH College of Engineering, JNTUH University, Hyderabad, India

Article Info

Article history:

Received May 18, 2021
Revised Jan 15, 2022
Accepted Feb 5, 2022

Keywords:

Built-in self-test
Fault-map unit
March SS algorithm
Memory built-in self-repair
SRAM
System-on-chip

ABSTRACT

The technology shrinkage and the increased demand for high storage memory devices in today's system on-chips (SoCs) has been the challenges to the designers not only in the design cycle but also to the test engineers in testing these memory devices against the permanent faults, intermittent and soft errors. Around 90% of the chip area in today's SoCs is being occupied by the embedded memories, and the cost for testing these memory devices contributes a major factor in the overall cost and the time to market. This paper proposes a strategy to develop a word-oriented March SS algorithm-basedmemory built-in self-test (MBIST), which is then applied for memory built-in self-test and repair (MBISTR) strategy. The implementation details for 1 KB of single-port static random-access memory (SRAM) depict that the modified March-SS algorithm based MBISTR-enabled SRAM facilitates self-test and self-repair of embedded memories with a marginal hardware overhead (<1%) in terms of look up tables and slice registers when compared to that of standard SRAM.

Corresponding Author:

Gobinda Prasad Acharya
Department of Electronics and Communication Engineering, Sreenidhi Institute of Science and Technology
501301, Hyderabad, Telangana, India
Email: gprasad04@gmail.com

1. INTRODUCTION

The system on-chip (SoC) architecture primarily consists of a processor core with single or many processing elements (cores) and embedded memories. The SoCs mainly consists of logic cores, memory partners, interconnects and I/O peripherals. According to the survey of the semiconductor industry association (SIA), embedded memories occupy more than 90% of the silicon area in modern day SoCs. The logic core components in SoCs are tested using logic built-in self-test (LBIST) strategies. i) The LBISTcan be classified into hardware-BIST. ii) software based self-test (SBST). The LBIST strategies are though capable of detecting manufacturing (permanent) faults, these techniques are not effective in handling transient faults which may occur when the system is deployed on the field. The Berger code based concurrent testing [1] is capable of detecting the transient faults.

The requirement for increased data storage (embedded memories) in today's complex SoCshas driven the designers to integrate millions of transistors on a silicon wafer by minimizing the device (transistor) size. The miniaturization in very large-scale integration (VLSI) technology has increased the parasitic effects and manufacturing defects that includebridges, opens, and shortsas well. The cross talks, process variation, parasitics, and short channel effects have introduced newer faults and fault models in high density semiconductor memories affecting the system behavior and performance.

The performance and reliability of SoCs depend hugely on the fault patterns and testing strategies for built-in memories. The high-density static random-access memorys (SRAMs)generally comprises of higher number of manufacturing defects per unit die-area which may result into lower yield as contrastto other glue logic. In addition, the cost of memory testing also increases with the increased memory density. With the availability of precise fault modeling and effective memory built-in self-test strategies, an improvement both in fault coverage and in the yield for embedded memories in today's SoCsis quite possible. This research work aims to develop an effective test methodology for self-testing and repairing of embedded memories in SoCs.

Fault modelingand test strategies for semiconductor memories, is the logical fault models are used to map or model the physical defects (opens, shorts, and bridges) in a circuit to logical faults. The classical fault models used for logic circuits and glue logic are not effective to model memory faults. The functional fault models are the best to model memory faults outlined in [2]–[4].

The effectiveness of a testmethodologyis measured using the metrics: fault coverage and test time. The VLSI test engineers have developed effective fault-diagnosis algorithms by targeting the memory fault models. These algorithms are aimed to improve the fault coverage and also to minimizethe test time. The test strategies for semiconductor memories are categorized into i) classical test methods and ii) march algorithm-based memory built-in self-test (MBIST) methods.

Classical test approaches, the classical test approaches for memory testing presented in the literature [4] suffer with low fault coverage and/or higher test time. March algorithms based MBIST techniques facilitate self-test memories without the need of external test hardware [5]–[10]. The March algorithms are more suitable for fault diagnosis and self-testing of regular 2-D memory architectures. March algorithms [11]–[14] performs March operations (memory write, and memory read operations) in a predefined sequence (ascending or descending order) of memory addressing. Each March operation could be: i) Write 0 (W0) into a memory cell. ii) Write 1(W1) into a memory cell. iii) Reading for '0' (R0) from the addressed memory cell. iv) Reading for '1' (R1) from the addressed memory cell in March algorithms, the following notations are used:

⇑ accessing the memory in an increasing order of its addresses (i.e., from 0 to 2^n-1)
⇓ accessing the memory in a decreasing order of its addresses (i.e., from 2^n-1 to 0)
⇕ accessing the memory in any order of its addresses

The commonly used March algorithms are summarized and compared in Table 1. These algorithms are compared with respect to the following parameters: the required number of March elements and March steps, test sequence, and the fault detection capability. As summarized in the Table 1, the March SS algorithm involves two consecutive Read operations during test sequencing and hence can additionally detect read destructive faults (RDFs) apart from the majority of other faults. This advantage has been the motive behind the selection of March SS algorithm this work though the test complexity (number of March elements and March steps) is higher when compared to other March algorithms.

Table 1. Comparison of various March algorithms

S. No	March Algorithm	No. of March elements	No. of March steps	Test Sequence	Fault Detection capability
1	MATS	4N	3	{⇕W0, ⇕ (R0, W1), ⇕R1}	SAFs, ADFs
2	MATS+	5N	3	{⇕W0, ⇑ (R0, W1), ⇓ (R1,W0)}	SAFs, ADFs
3	MATS++	6N	3	{⇕W0, ⇑ (R0, W1), ⇓ (R1,W0, R0)}	SAFs, ADFs, TFs, CFs
4	March A	15N	5	{⇕W0, ⇑ (R0,W1,W0,W1), ⇑ (R1,W0,W1), ⇓ ((R1,W0,W1, W0), ⇓ (R0,W1, W0)}	SAFs, ADFs, TFs
5	March B	17N	5	{⇕W0, ⇑ (R0,W1,R1,W0,R0,W1), ⇑(R1,W0,W1) ⇓((R1,W0,W1,W0), ⇓ (R0,W1,W0)}	SAFs, ADFs, TFs, CFs
6	March C	11N	7	{⇕W0, ⇑ (R0,W1), ⇑ (R1,W0), ⇕R0, ⇓ (R0,W1), ⇓ (R1,W0), ⇕R0 }	SAFs, ADFs, TFs, Some CFs
7	March X	6N	4	{⇕W0, ⇑ (R0,W1), ⇓ (R1,W0), ⇕R0 }	CFs
8	March Y	8N	4	{⇕W0, ⇑ (R0,W1, R1), ⇓ (R1,W0, R0), ⇕R0 }	SAFs, ADFs, TFs, CFs
9	March SR+	18N	6	{⇓W0, ⇑ (R0,R0,W1,R1,R1,W0,R0), ⇓R0, ⇑W1, ⇓ (R1,R1,W0,R0,R0,W1,R1), ⇑R1}	SAFs, ADFs, TFs, CFs
10	March SS	22N	6	{⇕W0, ⇑ (R0,R0,W0,R0,W1), ⇑ (R1,R1,W1,R1,W0), ⇓(R0,R0,W0,R0,W1) ⇓ (R1,R1,W1,R1,W0), ⇕R0}	SAFs, ADFs, TFs, CFs, RDFs

The March algorithms based MBIST techniques available in the literature are based on bit-access. As the size of memory increases, the number of March operations also increases thereby increasing the test time. The test time can be reduced if the March elements operate on memory words. This paper is focused on the adaptation of March SS algorithm for word-oriented embedded memories.

2. PROPOSED WORD-ORIENTED MARCH-SS ALGORITHM FOR MEMORY BUILT-IN SELF-TEST AND REPAIR

All the March algorithms found in the literature operate March operations on each memory cell (i.e. at bit-level) in ascending or descending or any order of memory addressing. This bit-wise access of meory cells during MBIST increases the number of March operations which results into enormous delayin test time for high density memories. In this work, the bit-oriented March SS algorithm has been modified into word-oriented (with word length of 8 bits) March SS algorithmso that the entire row (word) of the memory under test (MUT) can be accessed during March operation. The required test/reference patterns, denoted as W0, W1, W7, R0, R1, R7 that can detect all possible faults in the MUTare depicted in Table 2.

The test sequencing for the proposed word-oriented March SS algorithm is given in Table 3. During each memory read operation, the read out data from the MUT is compared with the reference pattern {Ri (i=0,1,2,… 7)} in the output response analyzer (ORA) module. The ORA module detects the presence of fault, if any in the addressed location. The modified word-oriented March SS algorithm has a test complexity of 82xN1 to complete the memory test, where N1 is the size of MUT in terms of the number of address locations.

SAFs, TFs, and ADFs which leads to the majority of the memory faults can be detected with two test patterns W0 and W1. The work presented in this paper focuses on the development of test architecture and analysis of simulation work for these two test patterns only. The number of word-level March operations for these two test patterns in the proposed modified word-oriented March SS algorithm is 22xN1.

Table 2. Test patterns for memory write andreference patterns for memory read operations

S.No.	Test Pattern	Notation	Reference Pattern	Notation	Targetted Faults
1	0 0 0 0 0 0 0 0	W0	0 0 0 0 0 0 0 0	R0	SAFs, ADFs, TFs
2	1 1 1 1 1 1 1 1	W1	1 1 1 1 1 1 1 1	R1	
3	0 0 0 0 1 1 1 1	W2	0 0 0 0 1 1 1 1	R2	CFs, NPSFs
4	1 1 1 1 0 0 0 0	W3	1 1 1 1 0 0 0 0	R3	
5	0 0 1 1 0 0 1 1	W4	0 0 1 1 0 0 1 1	R4	
6	1 1 0 0 1 1 0 0	W5	1 1 0 0 1 1 0 0	R5	
7	0 1 0 1 0 1 0 1	W6	0 1 0 1 0 1 0 1	R6	
8	1 0 1 0 1 0 1 0	W7	1 0 1 0 1 0 1 0	R7	

Table 3. Test sequence for the proposed word-oriented March SS algorithm

March Step	Test Sequence	March Step	March Sequence
1	$\Updownarrow$ W0	10	$\Downarrow$ (R0,R0,W0,R0,W1)
2	$\Uparrow$ (R0,R0,W0,R0,W1)	11	$\Downarrow$ (R1,R1,W1,R1,W2)
3	$\Uparrow$ (R1,R1,W1,R1,W2)	12	$\Downarrow$ (R2,R2,W2,R2,W3)
4	$\Uparrow$ (R2,R2,W2,R2,W3)	13	$\Downarrow$ (R3,R3,W3,R3,W4)
5	$\Uparrow$ (R3,R3,W3,R3,W4)	14	$\Downarrow$ (R4,R4,W4,R4,W5)
6	$\Uparrow$ (R4,R4,W4,R4,W5)	15	$\Downarrow$ (R5,R5,W5,R5,W6)
7	$\Uparrow$ (R5,R5,W5,R5,W6)	16	$\Downarrow$ (R6,R6,W6,R6,W7)
8	$\Uparrow$ (R6,R6,W6,R6,W7)	17	$\Downarrow$ (R7,R7,W7,R7,W0)
9	$\Uparrow$ (R7,R7,W7,R7,W0)	18	$\Updownarrow$ (R0)

2.1. Modified word-oriented March SS algorithm based MBIST architecture

A 1KB (1024x8 bit) single-port RAM (SPRAM) has been considered as MUT in this work. The MBIST architecture, depicted in Figure 1 consists of i) MUT, ii) a 2x1 Multiplexer (MUX), which selectsappropriate address, data, write and read control signals during the normal and test mode of operation, iii) test pattern generator (a REG file consisting of two registers holding 8-bit test patterns W0and W1), and iv) address sequencer (a 10-bit binary up-down counter). The entire operation of MBIST architecture is controlled by a MBIST controller which is a finite state machine (FSM). The MUT can be operatedeither in normal mode (TM=0) or in test mode (TM=1). For normal mode, the MUX selects the input/output functional data, memory address lines, memory write and memory read control signals so that normal functional operation (memory write and memory read) is carried out. During test mode of operation, the multiplexer selects the test patterns (W0 or W1from the REG file), memory address generated by address sequencer, and memory read/write control signals based on address sequencing.

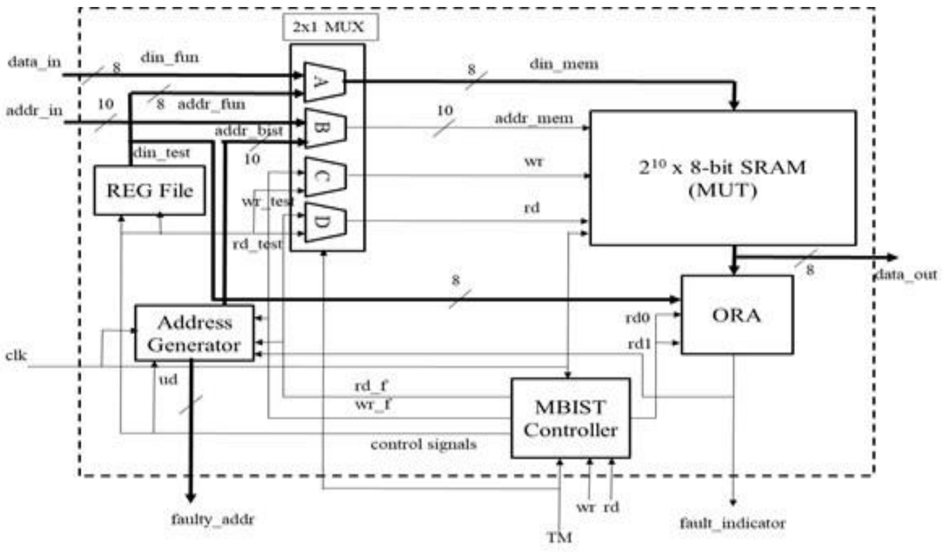


Figure 1. March SS based MBIST architecturefor word-oriented memory

2.2. Design of MBIST controller

The proposed MBIST controller whose FSM diagram is depicted in Figure 2 performs the following operations: i) It activates the up-down counter which generates the addresses for the MUT in a pre-defined sequence governed by March SS algorithm, ii) It asserts the memory write operation in the addressed memory location to write the selected test pattern (W0 or W1), and iii) It asserts the memory read operation from addressed memory location. After the memory read operation, the ORA module compares the read out data from the memory with the reference pattern to detect any faults in the addressed memory location.

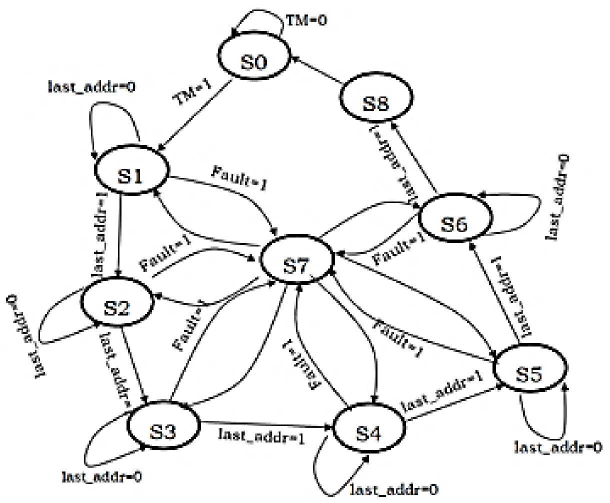


Figure 2. State diagram of the FSM based MBISTcontroller

2.3. Memory built-in self-test and repair (MBISTR) strategy

The memory built-in self-repair (MBISR) strategies found in the literatures [15]-[26] are capable of self-repairing the faulty embedded memory modules with additional hardware in terms of redundant memory array (additional rows and/or columns). The architecture of MBISTR strategy [27] proposed in this work is depicted in Figure 3. The architecture consists of two main modules; i) MBIST-enabled SRAM, and ii) a self-repair unit, which primarily consists of redundant memory array (RMA). In the test mode of operation, the faulty locations in MUT are stored in a faulty address memory (FAM) array addressed through FM_addr generated using fault-map (FM) address generator which is a mod-16 counter incremented every time a faulty location in MUT is detected. These faulty addresses are mapped to the RMA i.e., the multiplexer selects output data from the RMA to retrieve fault-free data.The size of FAM and RMA is the key for hardware overhead in the MBISTR architecture. Considering the hardware overhead in MBISTR architecture, a FAM of size 2^4x 10 is chosen which is capable of storing a maximum of sixteen 10-bit faulty addresses of MUT. All the 16 faulty addresses of MUT can be mapped with 2^4x8 RMA. At the end of memory test, the MUT will return to functional mode with FAM consisting of all the faulty addresses from RMA.

The FSM diagram for the MBISTR controller which controls all the operation of MBISTR architecture is depicted in Figure 4. At ‘S0’state, the input address to the memory is checked in FAM using the fault-map unit (FMU). When a match is found, the FMU asserts addr_matched signal to 1 indicating the addressed location in the primary memory is faulty. The MBISR controller then goes to the state ‘S1’, wherein, the subsequent memory (write or read) operations to these faulty addresses are switched into the RMA by asserting wr_rma or rd_rma signals and then the controller then enters into ‘S2’ state. At the end of memory access cycle, the controller returns to ‘S0’state.

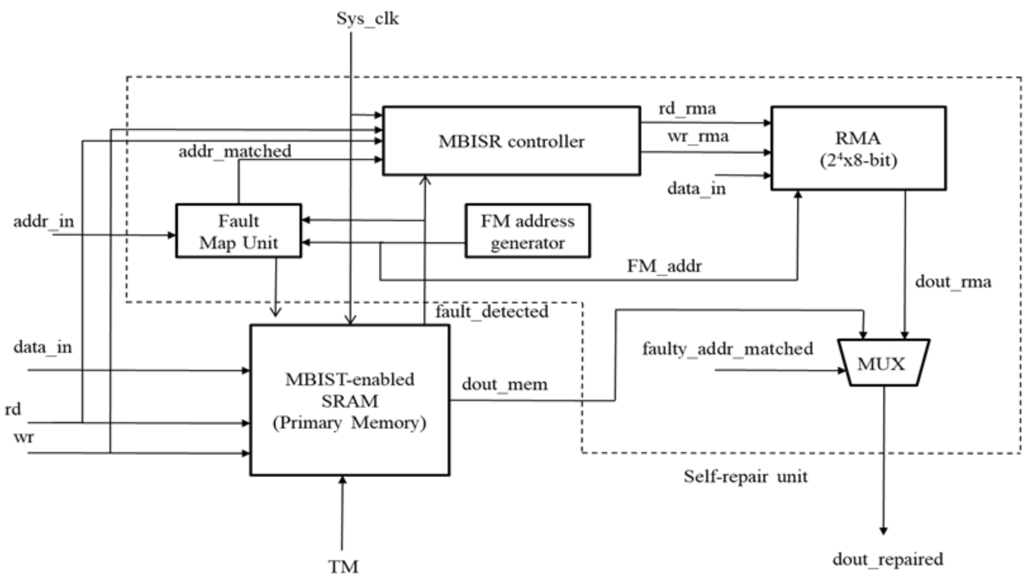
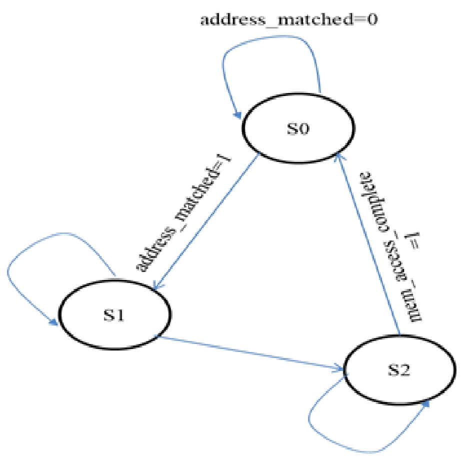


Figure 3. Proposed architecture of MBISTR



State	Activity	Description
S0	Initialization	Checks the input address with the entries in FAM until **address_matched** is asserted by FMU. The controller moves to state S1 when it is asserted.
S1	Self-repair	Access fault-mapped address from RMA by asserting **wr_rma** and **rd_rma** signals. Issues **FM_addr** to RMA.
S2	Repair done	The controller returns back to S0 state when the memory access is completed.

Figure 4. Typical state diagram for FSM based MBISR controller

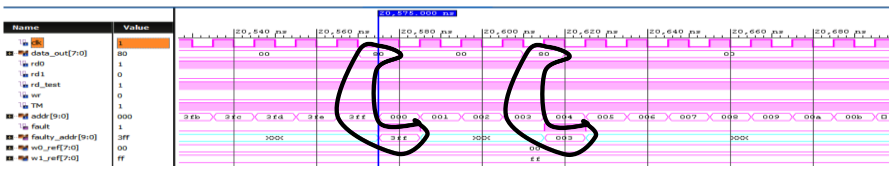
3. RESULTS AND DISCUSSION

The simulation in this work has been carried out in Xilinx Vivado 2017.4 environment using Verilog HDL. Xilinx SynthesisTools has been used to synthesize the proposed March-SS algorithm for word-oriented MBIST and MBISTR. The effectiveness of this method is presented and discussed in the following sub-sections.

3.1. Simulation results and implementation details of MBIST

During test mode of operation, the 2x1 MUXat the input side of the MUT selects the memory address from the address generator. It also selects the test patterns from REG fileandmemorywrite and memory read control signals as per the test sequencing defined by the proposed word-oriented March SS algorithm. The functional simulation for the proposed 1 KB MBIST controller is carried out at 100 KHzclock frequency. The proposed methodology has a test complexity (number of memory operations)of 322*N1 which results into a simulation delay of 225.50 ms.

The fault simulationin this work is carried out by injecting stuck-at-1 (SA1)faults at the most significant bit (MSB) position of memory locations 3FFH and 003H. These two faults are detected during the test mode of MUT when R0 operationis carried outwhen the MBIST controller is in the state ‘S2’. When these faults are injected, the read outdata for R0 operation following W0 operation will be 80H instead of 00H. The faulty locations (faulty_addr[9:0]) are stored for possible self-repair activity, discussed in the next section. The detection of memory faults at address locations 3FFH and 003Hhas been depicted in the simulation outcome shown in Figure 5.



Fault injected: S@1 fault at most significant bit (MSB) position of address locations 4’h3FF and 4’h003.
Inputs
TM=1 (**MBIST operation)**
rd_test=1, rd0=1 (**Memory read operation – R0)**
data_out: 4’h80 (Output of read data from memory to the input of ORA)
Outputs

Figure 5. Illustration of fault detection using word-oriented March SS algorithm based MBIST technique

The 1 KB single-port RAM with MBIST capability is implemented on 7-series Zynq Field programmable gate arrays (FPGA) (Xc7z020clg484-1). The hardware utilization for 1KB SPRAM without and with MBIST hardware is compared and presented in Table 4. The results obtained in the experiment show that ainsignificant (less than 3%) hardware overhead (in terms of LUTs and slice registers) enables self-testing.

3.2. Simulation results and implementation details of MBISTR

The synthesizable register transfer level (RTL) code for MBISTR is written using verilog hardware description language (HDL) and implemented in 7-series Zynq FPGA(Xc7z020clg484-1). After the memory self-test and fault mapping process, the MUT turns into normal (functional) mode of operation. The design implementation of the proposed word-oriented March SS algorithm based MBISTR and its functional verification has been demonstrated in this section. The hardware overhead in terms of self-repair unit introduces the faulttolerance in the SRAM.

In this work, to illustrate the fault simulation capability of the proposed MBISTR architecture, a SA1 fault is inserted at MSB position of memory location addressed at 000H, 003Hand 3FFH. In the presence of this fault, the data read out from these addresses’ values will be 80Hagainsta test pattern of 00H written. The faulty addresses are stored in FAM as depicted in Figure 6. The self-repairing ability of the proposed architectureof MBISTR is demonstrated through the simulation outcome shown in Figure 7. The presence of SA1 fault at MSB position of address location (003H) has caused an erroneous read out data from memory to be data_out=D5H whenaninput data (data_in=55H) is written at this address. The correct data (Dout_repaired=55H) has been retrieved from the RMA. The hardware utilization for a single-port RAM (SPRAM), MBIST-enabled SPRAM, and MBISTR-enabled SPRAM is summarized in Table 4, which shows that the MBISTR-enabled SPRAM though requires a marginal (less than 1%) hardware overhead (LUTs and slice registers), introduces the fault-tolerance in the memory.

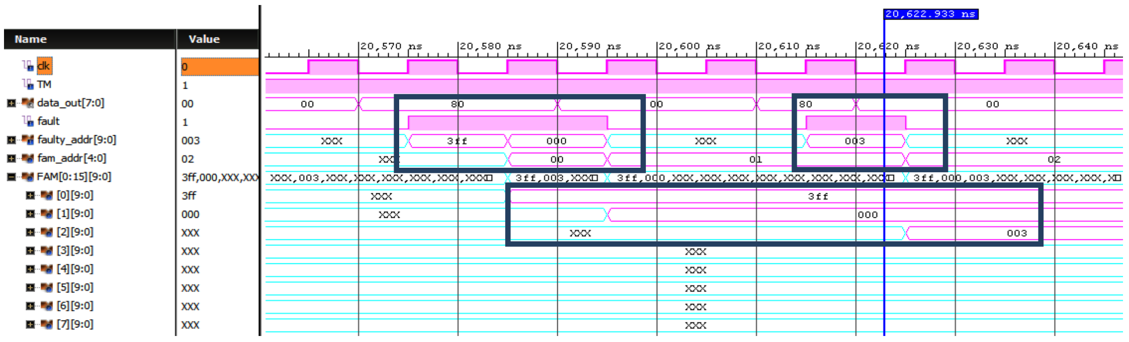


Figure 6. Simulation outcome demonstrating fault detection and fault-mapping

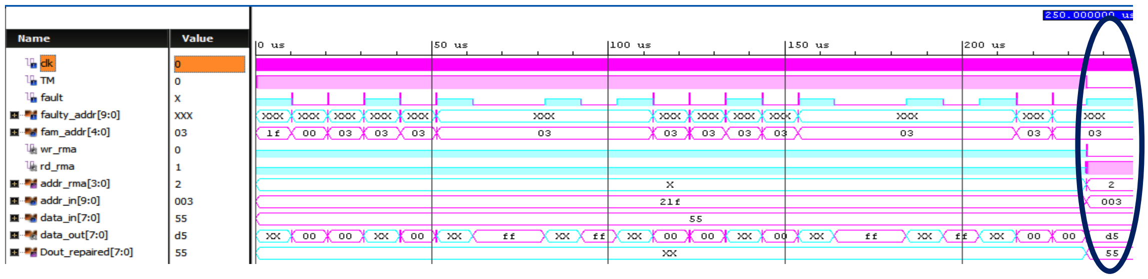


Figure 7. Simulation result of MBISTR for fault detection and self-repair

Table 4. Summery of hardware utilization of 1KB standard SPRAM, MBIST and MBISTR

S. No	Resource	Available	Standard SPRAM		MBIST-enabled SPRAM		MBISTR-enabled SPRAM	
			Utilization	%	Utilization	%	Utilization	%
1	Slice LUTs	17600	3516	19.98	3684	20.93	3715	21.07
2	Slice Registers	35200	8197	23.29	8256	23.45	8281	23.52
3	IOs	102	29	28.43	41	42.16	45	44.18
4	F7 Muxes	8800	1052	12.36	1052	12.36	1052	12.36
5	F8 Muxes	4400	472	12.36	472	12.36	472	12.36

4. CONCLUSION

This paper discusses a Memory Built-In Self-Test and Repair methodology based on a word-oriented March SS algorithmwhich is derived from the original bit-oriented March SS algorithm. The modified word-oriented March SS algorithm requires 22 word-level March operations to detect majority of memory faults i.e., SAFs, ADFs. The test complexity of the proposed architecture has been 22xN1 (=22,528) word-level operations for 1 KB (2^{10}x8) SPRAM, as compared to 22xN1x8 (=1,80,224) bit-level operations. The hardware utilization of MBISTR methodology requires a hardware overhead (LUTs and slice registers) of ~1% as compared to that of MBIST-enabled SRAM. This marginal hardware overhead has introduced a fault-tolerance in the embedded memories improving the reliability.

REFERENCES

[1] G. P. Acharya and M. A. Rani, "Berger code based concurrent online self-testing of embedded processors," *Journal of Semiconductors*, vol. 39, no. 11, p. 115001, Nov. 2018, doi: 10.1088/1674-4926/39/11/115001.

[2] R. Dekker, F. Beenker, and L. Thijssen, "Fault modeling and test algorithm development for static random access memories," in *Digest of Papers - International Test Conference*, 1988, pp. 343–352, doi: 10.1109/test.1988.207820.

[3] A. J. Van De Goor, "Using march tests to test SRAMs," *IEEE Design & Test of Computers*, vol. 10, no. 1, pp. 8–14, Mar. 1993, doi: 10.1109/54.199799.

[4] A. K. Sharma, *Semiconductor Memories: Technology, Testing, and Reliability*. Hoboken, New Jersey: John Wiley & Sons, 1997.
[5] K. Zarrineh and S. J. Upadhyaya, "On programmable memory built-in self test architectures," in *Proceedings -Design, Automation and Test in Europe, DATE*, 1999, pp. 708–713, doi: 10.1109/DATE.1999.761207.
[6] N. Z. Haron, S. A. M. J. Yunus, A. H. A. Razak, and M. Y. I. Idris, "Modeling and simulation of finite state machine memory built-in self test architecture for embedded memories," in *2007 Asia-Pacific Conference on Applied Electromagnetics Proceedings, APACE2007*, Dec. 2007, pp. 1–5, doi: 10.1109/APACE.2007.4603901.
[7] S. Priya, S. Hazra, B. Chakraborty, and M. Dalui, "A cellular automata based BIST for detecting NPSFs in high speed memories," in *ACM International Conference Proceeding Series*, Feb. 2018, pp. 306–311, doi: 10.1145/3185089.3185133.
[8] J. Park and K. W. Kwon, "A Built-In Self Test Compensating Process-Voltage Variation in Data Paths of High Performance DRAMs," in *2018 IEEE 10th International Memory Workshop, IMW 2018*, May 2018, pp. 1–4, doi: 10.1109/IMW.2018.8388779.
[9] K. J. Lee, B. R. Chen, and M. A. Kochte, "On-Chip Self-Test Methodology With All Deterministic Compressed Test Patterns Recorded in Scan Chains," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 2, pp. 309–321, Feb. 2019, doi: 10.1109/TCAD.2018.2808241.
[10] R. Zeli, R. Silveira, and Q. Qureshi, "SoC Memory test optimization using NXP MTR solutions," in *LATS 2019 - 20th IEEE Latin American Test Symposium*, Mar. 2019, pp. 1–5, doi: 10.1109/LATW.2019.8704566.
[11] 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, RAICS 2013*, Dec. 2013, pp. 199–204, doi: 10.1109/RAICS.2013.6745473.
[12] S. Hamdioui, A. J. Van De Goor, and M. Rodgers, "March SS: A test for all static simple RAM faults," in *Records of the IEEE International Workshop on Memory Technology, Design and Testing*, 2002, vol. 2002-January, pp. 95–100, doi: 10.1109/MTDT.2002.1029769.
[13] M. Parvathi, "Modified March C-With Concurrency in Testing for Embedded Memory Applications," *International Journal of VLSI Design & Communication Systems*, vol. 3, no. 5, pp. 43–51, Oct. 2012, doi: 10.5121/vlsic.2012.3504.
[14] D. Youn, T. Kim, and S. Park, "A microcode-based memory BIST implementing modified march algorithm," in *Proceedings of the Asian Test Symposium*, 2001, pp. 391–395, doi: 10.1109/ats.2001.990315.
[15] D. K. Bhavsar, "Algorithm for row-column self-repair of RAMs and its implementation in the Alpha 21264," in *IEEE International Test Conference (TC)*, 1999, pp. 311–318, doi: 10.1109/test.1999.805645.
[16] T. W. Tseng *et al.*, "A built-in self-repair scheme for multiport RAMs," in *Proceedings of the IEEE VLSI Test Symposium*, May 2007, pp. 355–360, doi: 10.1109/VTS.2007.4.
[17] V. Schöber, S. Paul, and O. Picot, "Memory Built-In Self-Repair using redundant words," in *IEEE International Test Conference (TC)*, 2001, pp. 995–1001, doi: 10.1109/TEST.2001.966724.
[18] C. T. Huang, C. F. Wu, J. F. Li, and C. W. Wu, "Built-In Redundancy Analysis for Memory Yield Improvement," *IEEE Transactions on Reliability*, vol. 52, no. 4, pp. 386–399, Dec. 2003, doi: 10.1109/TR.2003.821925.
[19] R. K. Sharma and A. Sood, "Modeling and simulation of multi-operation microcode-based built-in self test for memory fault detection and repair," in *Proceedings - IEEE Annual Symposium on VLSI, ISVLSI 2010*, Jul. 2010, pp. 381–386, doi: 10.1109/ISVLSI.2010.88.
[20] T. J. Chen, J. F. Li, and T. W. Tseng, "Cost-efficient built-In redundancy analysis with optimal repair rate for RAMs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 6, pp. 930–940, Jun. 2012, doi: 10.1109/TCAD.2011.2181510.
[21] T. Li, Y. Han, X. Liang, H. H. S. Lee, and L. Jiang, "Fault clustering technique for 3D memory BISR," in *Proceedings of the 2017 Design, Automation and Test in Europe, DATE 2017*, Mar. 2017, pp. 560–565, doi: 10.23919/DATE.2017.7927050.
[22] K. Cho, Y. W. Lee, S. Seo, and S. Kang, "An efficient built-in self-repair scheme for area reduction," in *Proceedings - International SoC Design Conference 2017, ISOCC 2017*, Nov. 2018, pp. 105–106, doi: 10.1109/ISOCC.2017.8368791.
[23] A. S. Syed, D. E. Rani, and M. A. Ahmed, "Embedded Memory Test Strategies and Repair," *International Journal of Engineering*, vol. 30, no. 6, pp. 839–845, 2017, doi: 10.5829/ije.2017.30.06c.03.
[24] T. Ni, H. Chang, Y. Yao, X. Li, and Z. Huang, "A Novel Built-In Self-Repair Scheme for 3D Memory," *IEEE Access*, vol. 7, pp. 65052–65059, 2019, doi: 10.1109/ACCESS.2019.2917195.
[25] H. Lee, D. Han, S. Lee, and S. Kang, "Dynamic built-in redundancy analysis for memory repair," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 10, pp. 2365–2374, Oct. 2019, doi: 10.1109/TVLSI.2019.2920999.
[26] P. Papavramidou and M. Nicolaidis, "Iterative Diagnosis Approach for ECC-Based Memory Repair," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 2, pp. 464–477, Feb. 2020, doi: 10.1109/TCAD.2018.2887052.
[27] G. P. Acharya, "Fault-Tolerance using Concurrent Test Procedure for Multi-core Processing Elements," Sreenidhi Institute of Science and Technology, 2021.

BIOGRAPHIES OF AUTHORS

Gobinda Prasad Acharya received the B. Tech (ECE) degree in the year 2004 from JNT University, Hyderabad, T.S., India. He received his Masters Degree in Engineering (Digital systems) from Osmania University, Hyderabad, and Ph.D. Degree from JNTUH University, Hyderabad. He has 17+ years of teaching and research experience and is presently working as an Associate Professor in the Department of ECE at Sreenidhi Institute of Science and Technology, Hyderabad, India. His area of research interests includes Fault Tolerant System, VLSI Design & Testing, and Embedded systems. He can be contacted at email: gpacharya@sreenidhi.edu.in.

Muddapu Asha Rani did her B.E from Osmania University, Hyderabad during 1986-90, completed her M.Tech and Ph.D from JNTU Hyderabad in 1997 and 2008 respectively. Presently she is working as Professor in the Department of Electronics and Communication Engineering, JNTUH College of Engineering, Kukatpally, Hyderabad. She has 30 years of teaching and research experience. Her research interest is Fault Tolerant System Design, Design for Testability, Embedded Systems Design, and VLSI Design. She has sucessfuly guided 5 scholars towards their Ph. D. Degree. She can be contacted at email: ashajntu1@jntuh.ac.in.

Ganjikunta Ganesh Kumar completed his B.Tech and M.Tech in Electronics and Communication Engineering from Jawaharlal Nehru Technological University (JNTU) - Anantapur, Anantapur, India in the year 2009 and 2011, respectively. He received the Ph.D. degree in Electronics and Communication Engineering from BITS-Pilani, Hyderabad Campus, in 2019. Presently he is working as Assistant Professor in the Department of Electronics and Communication Engineering at Sreenidhi Institute of Science and Technology, Hyderabad, India. His areas of research interests are VLSI arithmetic circuits and DSP. He can be contacted at email: ganeshkumarg@sreenidhi.edu.in.

Lavanya Poluboyina completed B.Tech. in E.C.E. from Vignan’s Engineering College, JawaharlalNehruTechnological University, Hyderabad, Andhra Pradesh, India in 2004 and received M.Tech. degree in Digital Systems and Computer Engineering from University College of Engineering, JNTUH, Hyderabad, Telangana, India in the year of 2009. She has teaching experience of more than 17 years and presently working as Assistant Professor in the department of E.C.E., Sreenidhi Institute of Science and Technology, Hyderabad, Telangana, India. Her areas of interest are Wirelessand Ad-hoc Networks; Routing in Wirelessand Ad-hoc Networks. She is a Member of IEEE. She can be contacted at email: lavanyap@sreenidhi.edu.in.