International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 12 (2017) pp. 3023-3028

CHECKERMARC: A Modified Novel Memory-Testing Approach for Bit-Oriented SRAM

Aditya Kumar Singh Pundir[1]
[1]Research Scholar, Department of Electronics & Communication Engineering, Poornima University Jaipur, Rajasthan, India.

Om Prakash Sharma[2]
[2]Professor, Department of Electronics & Communication Engineering, Poornima College of Engineering Jaipur, Rajasthan, India.

[1]ORCID: 0000-0002-6762-9263, [1]Scopus Author ID: 57191156111

Abstract

This paper presents a hybrid memory testing approach for SRAM by clubbing two existing conventional memory testing algorithms i.e. MARCH C- and Checker Board. The checkerboard algorithm is an improved version of zero-one memory testing algorithm. The main goal of this work is to adhere and use the best advantages that MARCH C- and Checker Board were offering, with a sharp eye on the tradeoff and analysis parameters such as area consumed, fault detection and diagnosis speed and power consumption and fault coverage. The proposed deterministic memory testing algorithm is design for targeted fault list that contains stuck at faults, stuck open faults, transition faults, data retention faults and coupling faults. The simulation results show that the fault coverage of CHECKERMARC is up to 97.17% as it not detects some coupling faults and data retention faults in the memory. The implementation of CHECKERMARC was being done on Virtex-5 FPGA and results confirms that in spite of having some area penalties the proposed algorithm has good diagnosis speed and up to 0.67 % better fault converge than conventional Checker Board and 0.8% better fault coverage from MARCH C- algorithm. The fault dictionary obtained from proposed algorithm is in the format of (row, column) which can be used for further repair actions.

Keywords: CHECKERMARC, CHECKER BOARD Algorithm, MARCH C- Algorithm, MBIST, Fault Dictionary, Injected Faults.

INTRODUCTION

The development of IC integration technologies leads to an extensive use of memories and buffers in different memory intensive applications. Therefore, probability of occurrence of fault in every single read and write operation is increased. There were many testing approaches [1], [2] and [3] were developed for efficient testing and diagnosis of fault [4], [5], [6], [7], [8] and [9]. However, all algorithms are not strengthened enough to detect all possible faults that may be present due to fabrication errors or environmental disturbance. Keeping this in mind and taking the possibility of development of efficient algorithm a hybrid memory testing algorithm is presented. The presented algorithm takes best suitable procedures from both conventional algorithms and provide a suitable framework for fault detection and diagnosis. The proposed algorithm is simulated first using MATLAB script for analysing its performance and fault coverage against injected fault covering stuck open faults [10], stuck at faults [11], transition faults [12] and coupling faults [13] and implemented using VHDL to analyse its hardware related parameters like area, delay and power. The transition faults and coupling faults are injected keeping assumption that only two cells exhibit this nature. The conventional algorithms are also simulated and implemented for same fault count and specifications and found that proposed algorithms have same fault coverage as that of MARCH C- algorithm [14] but having some nominal area penalty. The proposed algorithm also proves better as compared conventional checker board algorithm in terms of better fault coverage [15] however still having some area and speed related issues.

The paper structure is divided into four major sections. The introduction and relevance of proposed technique with the literature is presented in this section (Section I). the conventional checker board and MARCH C- algorithms are being discussed in Section II. The proposed hybrid memory testing algorithm is covered in Section-III. The Section-IV covers the working and simulation and implementation profile of proposed algorithm, Section-V covers the results and analysis of CHECKERMARC according to performance metrics taken. The overall conclusion of the paper is mentioned in Section-VI.

CONVENTIONAL CHECKER BOARD & MARCH C- ALGORITHMS

The process of Board-Checker algorithm is slightly different from Zero-One algorithm. In this algorithm user write one's and zero's at alternate locations of the memory instead of writing complementary pattern [16]. The pseudo code for this algorithm is mentioned in Figure (1). The fault coverage of the board checker algorithm includes SAFs if Address decoder is fault free, bridging faults, Some Transition Faults, Some Coupling faults [17], [18], [19].

A bit/word-oriented March C− algorithm is presented in [20] which consist of six marching elements named M0 to M5. As described in Figure (2). In M0, the notation means that any ascending and descending order is accepted in marching and (w0) means write '0' in memory from starting address location to ending address location. Similarly, for (w1) means writing '1' in memory in descending address order. The term (r0) or (r1) means reading of expected values as with the symbol r.

Algorithm 1: Conventional Checker Board

1. Start in any order ascending or descending.
2. For i=odd and j=even
   {
   Write '0' in ith Cell and '1' in jth Cell.
   Read expected values in ith Cell and jth Cell.
   If expectation fails, then copy Cell address.
   Complement all cells.
   Read expected values.
   If expectation fails, then copy Cell address
   }
3. Go to step 2 for repetition.

Figure 1: Pseudo Code for Checker Board Algorithm

The March algorithm says that writing of successive zero or one should be done according to marching element and order of marching. As mention in Figure (2) there are six march elements starting from M1 to M6. The M1 and M6 may follow any ordering of writing '1' or '0'. Whereas in Marching elements M2 to M6 the orders are being provided i.e. M2 and M3 is in ascending order and M4 and M5 is in descending order respectively. The M2, M3 and M4, M5 marching elements are same only ordering method is different. The fault coverage of March C- algorithm covers Stuck-at Faults, Address Faults, Transition Faults and some Coupling Faults [21], [22], [23], [24]. The next section presents the proposed hybrid memory testing algorithm.

Algorithm 2: Conventional March C-

$M1: \Updownarrow (w0);$
$M2: \Uparrow (r0, w1);$
$M3: \Uparrow (r1, w0);$
$M4: \Downarrow (r0, w1);$
$M5: \Downarrow (r1, w0);$
$M6: \Updownarrow (r0)$

Figure 2: Pseudo Code for March C-

PROPOSED HYBRID MEMORY TESTING ALGORITHM

The proposed algorithm is as shown in the Figure (3). It defines a hybrid marching technique to test SRAM. As in Figure (1), Checker board algorithm the pattern '0' or '1' is filled in memory with even and odd locations. Whereas in March C- algorithm according to March element M1 to M6 the patterns are successive read and write operations are performed. The proposed algorithm combines both techniques by reading and writing on even and odd locations according to marching patterns. The main benefit of marching at even and odd locations is that three fault profiles i.e. transition faults, coupling faults and bridging faults [25] can easily covered as compared to checkerboard algorithm.

Algorithm 3: Hybrid Proposed CHECKERMARC

$M1: \Updownarrow [(w0)_{i=even} \text{ and } (w1)_{i=odd}];$
$M2: \Uparrow \left[((r0)_{i=even} \text{ and } (r1)_{i=odd}), ((w1)_{i=even} \text{ and } (w0)_{i=odd})\right];$
$M3: \Uparrow \left[((r1)_{i=even} \text{ and } (r0)_{i=odd}), ((w0)_{i=even} \text{ and } (w1)_{i=odd})\right];$
$M4: \Downarrow \left[((r0)_{i=even} \text{ and } (r1)_{i=odd}), ((w1)_{i=even} \text{ and } (w0)_{i=odd})\right];$
$M5: \Downarrow \left[((r1)_{i=even} \text{ and } (r0)_{i=odd}), ((w0)_{i=even} \text{ and } (w1)_{i=odd})\right];$
$M6: \Updownarrow [(r0)_{i=even} \text{ and } (r1)_{i=odd}].$

Figure 3: Pseudo Code for Proposed CHECKERMARC

Another advantage of this algorithm is that it inherits the advantages of Checker board algorithm and March C- algorithm in terms of hardware easiness, high fault coverage

and low complexity. To complete the implementation and simulation analysis of proposed algorithm Section IV presents the base architecture and working of implemented MBIST module.

IMPLEMENTATION & WORKING OF HYBRID CHECKERMARC ALGORITHM

For the testing of proposed CHECKERMARC algorithm a memory built in self-test (MBIST) hardware structure for SRAM is developed, that consist of a MBIST controller to control the ordering of marching, marching element state sequences and even, odd trace of loop. A synchronous bit oriented memory that has four cases: 256x8x1, 1Kx8x1, 256x16x1 and 1Mx16x1. The error checking module that work is to check the output response according to mismatching of data output and the data available in the output response analyser. If data matching is different, then the concerned location is stored in the fault dictionary. The fault dictionary is having the details of addresses in terms of row and column of SRAM. This information is useful for further repairing solutions of the memory available in the fault dictionary. The format of fault dictionary is as shown in Figure (4).

The block diagram of proposed MBIST module is as shown in the Figure (5) which contains MBIST Controller, march sequence controller, SRAM, Comparator, Output response analyser, comparator and fault dictionary. Once the checking is completed and the comparator output confirms that the output of memory and data in ORA is same then memory is good for use. Otherwise corresponding address is being stored in the fault dictionary for doing repairing actions according to redundancy analysis algorithms [26].

Index →	0	1
Index ↓	**F_r**	**F_c**
0	**00**	**010**
1	**01**	**100**
…	**…**	**…**
n	**11**	**001**

Figure 4: Fault dictionary

The next section includes the results and outcome according to simulation and implementation profile.

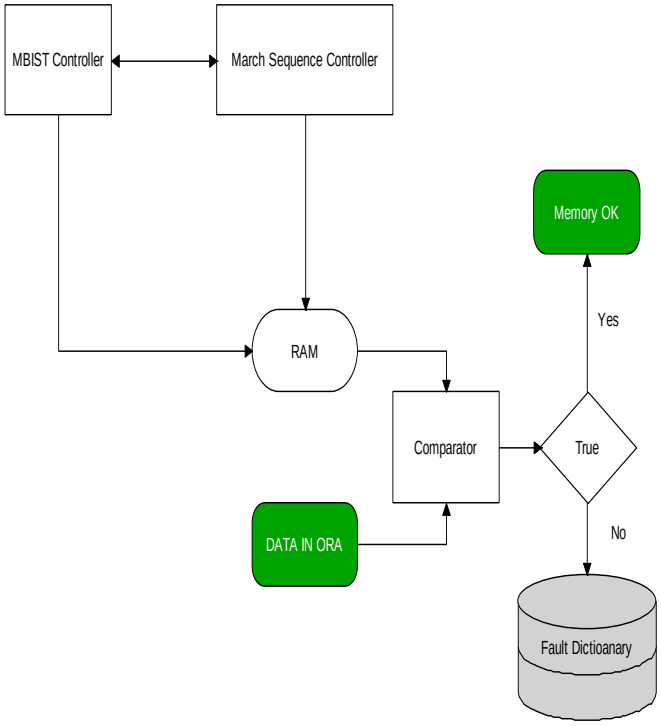

Figure 5: Block Diagram of proposed MBIST Structure.

RESULTS & DISCUSSION

The simulation and implementation profiles that was adopted in this paper is as shown in Table (1).

Table 1: Simulation and Implementation Profile

Environment/ Tool/ Fault Profile
VHDL- Virtex 5 Board/ Xilinx 12.1 (50 Random Faults) 10 SAFs, 10 SOFs, 10 Transition Faults 10 Coupling Faults 10 Bridging faults

The specifications that were taken for the simulation and implementation of three important memory testing algorithms is as shown in Table (1). There are four SRAM cases were included to test this algorithm as mentioned in the Table (2). So as shown in the Figure (6) for simplicity and fast simulation checker board algorithm is better than rest two algorithms. But for better fault coverage CHECKERMARC algorithm gives as equal as March C- algorithm. In spite of having some area overhead the proposed algorithm gives better results compared to both two conventional algorithms as shown in Figure (7).

Table 2: Testing algorithm comparison

S.No.	Algorithm Type	SRAM Profile	Test Time (ns)
1	Checker Board	256x8x1	35.79
		1Kx8x1	38.72
		256x16x1	32.45
		1Kx16x1	33.47
2	CHECKERMARC	256x8x1	36.12
		1Kx8x1	37.51
		256x16x1	30.5
		1Kx16x1	35.3
3	March C-	256x8x1	30.5
		1Kx8x1	31.93
		256x16x1	28.82
		1Kx16x1	29.38

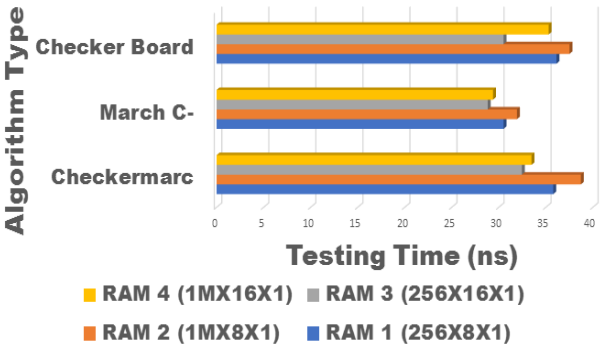

Figure 6: March Algorithms comparison

The Figure (8) shows the area overhead in terms of number of Slice registers that are used in logic design in FPGA. It shows that proposed algorithm is having nominal area overhead compared to both two conventional algorithms because it adopts hybrid testing methodologies used in both two algorithms.

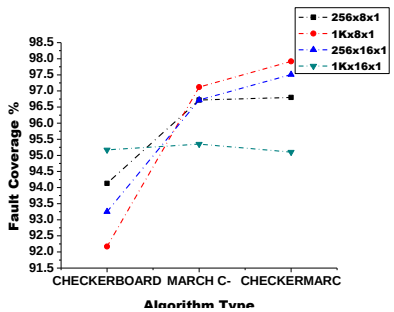

Figure 7: Fault coverage comparison between Testing algorithms and CHECKMARC

The maximum average delay comparison of three algorithms are displayed in Figure (9), it shows the maximum average delay of an input signal traverse from input to output according to critical path. It was found that proposed MBIST algorithm has outperform in maximum average delay parameter compared to checker board and march C- algorithms.

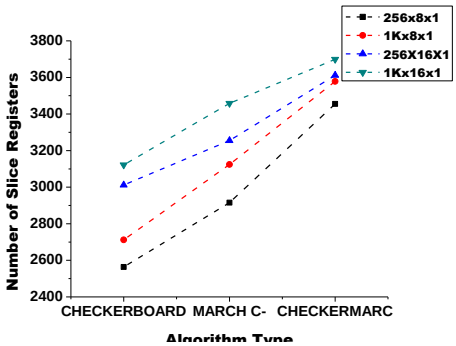

Figure 8: Area Comparison between Testing algorithms and CHECKMARC

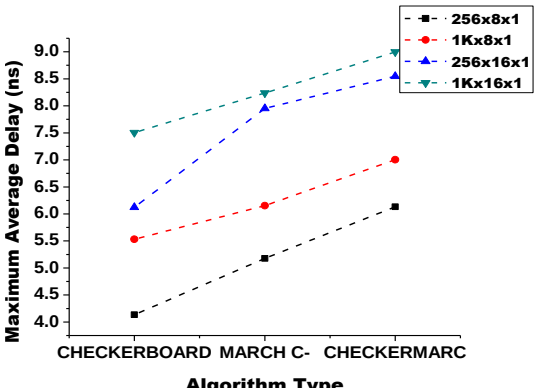

Figure 9: Maximum Average Delay comparison between Testing algorithms and CHECKMARC

CONCULSION

An improved version of checker-board and march c-algorithm is proposed by clubbing them together to adhere the advantages of both two memory testing methodologies. The proposed hybrid memory testing algorithm shows better results and performance in terms of fault coverage and hardware easiness with some nominal area overhead issues. The fault coverage obtained for presented algorithm is up to 97.17% and the implementation was done on Virtex-5 FPGA. The results confirm that in spite of having some area penalties the proposed algorithm has good diagnosis speed and up to 0.67 % better fault converge than conventional Checker Board and 0.8% better fault coverage from MARCH C- algorithm.

REFERENCES

[1] R. O. B. Dekker, F. Beenker, and L. Thijssen, "Static Random Access Memories," IEEE Trans. Comput. Des., vol. 9, no. 6, pp. 567–572, 1990.

[2] Ad J. Van De Goor ," Using March Tests to Test SRAMs", IEEE Design & Test of Computers, Vol. 10, no I, 8-14, 1993.

[3] [Ad J. Van De Goor et.al, 1994] Ad J. Van De Goor and Y. Zorian, "Effective March Algorithms for Testing Single-Order Addressed Memories," J. Electron. Test. Theory Appl., vol. 345, no. I, pp. 337–345, 1994.

[4] Pinaki Mazurnder, Kanad Chakraborty, "Testing and Testable Design of High-Density Random-Access Memories", Kluwer Academic Publishers, ISBN-13: 978- 1-4612-8632-5,1996.

[5] C. Wu, C. Huang, K. Cheng, and C. Wu, "Simulation-Based Test Algorithm Generation for Random Access Memories," IEEE Proc. 18th VLSI Test Symp., vol. I, no. I, pp. 291–296, 2000.

[6] Huang, J. Huang, and C. Wu, "A Programmable Built-In Self-Test Core for Embedded Memories," IEEE Des. Autom. Conf., vol. I, no. I, pp. 11–12, 2000.

[7] K. C. H.-P. L. Chih-Wea Wang, Chi-Feng Wu, Jin-Fu Li, Cheng-Wen Wu, Tony Teng, "A built-in self-test and self-diagnosis scheme for embedded SRAM," IEEE Proc. Ninth Asian Test Symp., vol. I, no. I, pp. 45–50, 2000.

[8] C. Huang, C. Wu, J. Li, and C. Wu, "Built-In Redundancy Analysis for Memory Yield Improvement," IEEE Trans. Reliab., vol. 52, no. 4, pp. 386–399, 2003.

[9] M. U. De. K. Gunavathi, Mr. K. Paramasivam, Ms. P. Subashini Lavanya, "A Novel BIST TPG for Testing of VLSI Circuits," IEEE Int. Conf. Ind. Andin. Syst., vol. I, no. August, pp. 8–11, 2006.

[10] H. Li, J. Mundy, A. Zaslavsky, and R. I. Bahar, "Therm ally-induced soft errors in nanoscale CMOS circuits," IEEE Int. Symp. Nanoscale Archit., pp. 62–69, 2008.

[11] Voyiatzis, "An ALU-Based BIST Scheme for Word-Organized RAMs," IEEE Trans. Comput., vol. 57, no. 5, pp. 577–590, 2008.

[12] Heather M. Quinn, Paul S. Graham, Michael J.Wirthlin, Brian Pratt, Keith S. Morgan, Michael P. Caffrey, and James B. Krone, "A Test Methodology for Determining Space Readiness of Xilinx SRAM-Based FPGA Devices and Designs," IEEE Trans. Instrum. Meas., vol. 58, no. 10, pp. 3380–3395, Oct. 2009.

[13] H.-C. Lu and J.-F. Li, "A programmable online/off-line built-in self-test scheme for RAMs with ECC," 2009 IEEE Int. Symp. Circuits Syst., pp. 1997–2000, May 2009.

[14] A.P. De Souza, F. M. De Assis, R. Carlos, and S. Freire, "A New Architecture of Test Response Analyzer Based on the Berlekamp – Massey Algorithm for BIST," IEEE Trans. Instrum. Meas., vol. 59, no. 12, pp. 3168–3173, 2010.

[15] Dong Wang, Ercegovac, Miloš D., Zheng, Nanning, "Design of High-Throughput Fixed-Point Complex Reciprocal/Square-Root Unit," IEEE Trans. Circuits Syst. II Express Briefs, vol. 57, no. 8, pp. 627–631, Aug. 2010.

[16] Parandeh-Afshar, A. K. Verma, P. Brisk, and P. Ienne, "Improving FPGA Performance for Carry-Save Arithmetic," IEEE Trans. Very Large Scale Integr. Syst., vol. 18, no. 4, pp. 578–590, Apr. 2010.

[17] C. Bolchini, A. Miele, and A. Sandionigi, "A Novel Design Methodology for Implementing Reliability-Aware Systems on SRAM-Based FPGAs," IEEE Trans. Comput., vol. 60, no. 12, pp. 1744–1758, 2011.

[18] Farah B and Mohammad Mansour, "Determining the Minimum Energy Operating Point for Embedded SRAM Memory," IEEE Trans. Very Large Scale Integr. Syst., vol. 1, no. 1, pp. 112–116, 2011.

[19] H.-C. Chung, Z.-Y. Chen, and P.-C. Chang, "Low power architecture design and hardware implementations of deblocking filter in H.264/AVC," 2011 IEEE Int. Conf. Consum. Electron., pp. 405–406, Jan. 2011.

[20] Mousumi Saha, Souvik Das and Biplab K Sikdar, "High Speed Hardware For March C–", International Symposium on Electronic System Design (ISED), Kolkata, pp. 145-147, 2012.

[21] G. Gyepes, D. Arbet, J. Brenkus, and V. Stopjaková, "Application of I DDT test towards increasing SRAM reliability in nanometer technologies," IEEE Trans. Very Large Scale Integr. Syst., pp. 1–4, 2012.

[22] H. Cao, M. Liu, H. Chen, X. Zheng, C. Wang, and Z. Wang, "Efficient Built-in Self-Repair Strategy for Embedded SRAM with Selectable Redundancy," IEEE Trans. Comput., vol. I, no. 2012, pp. 2565–2568, 2012.

[23] C. Hou and J. Li, "High Repair-Efficiency BISR Scheme for RAMs by Reusing Bitmap for Bit Redundancy," IEEE Trans. Very Large Scale Integr. Syst., vol. I, no. I, pp. 1–9, 2014.

[24] Filipek Michał, Mrugalski Grzegorz, Mukherjee, Nilanjan, Nadeau-dostie, Benoit Rajski, Janusz, "Low-Power Programmable PRPG With Test Compression Capabilities," IEEE Trans. Very Large Scale Integr. Syst., vol. 23, no. 6, pp. 1063–1076, 2014.

[25] Aditya Pundir, Dr. Om Prakash Sharma, "Fault Tolerant Reconfigurable Hardware Design Using BIST On SRAM: A Review" , International Conference on Engineering and Technology (ICET), Karpagam College of Engineering, Coimbatore, pp. 76-92, ISBN: 978-1-5090-3213-6, December 16-17, 2016.

[26] J. Kim, W. Lee, K. Cho, S. Kang, and S. Member, "Hardware-Efficient Built-In Redundancy Analysis for Memory With Various Spares," IEEE Trans. VERY LARGE SCALE Integr. Syst., vol. 25, no. 3, pp. 844–856, 2017