International Journal of Engineering

Journal Homepage: www.ije.ir

Embedded Memory Test Strategies and Repair

M. Altaf Ahmed*[a], D. Elizabath Rani[a], S. A. Sattar[b]

[a] *Department of ECE, GITAM Institute of Technology, GITAM University, Visakhapatnam, India*
[b] *Department of ECE, Royal Institute of Technology & Science, A. P. India*

PAPER INFO

Paper history:
Received 13 July 2016
Received in revised form 24 April 2017
Accepted 24 April 2017

Keywords:
Embedded Memory
Self-testing
Memory Built-in self-repair
System on Chip
Memory Test Algorithm
Yield Improvement

ABSTRACT

The demand of self-testing proportionally increases with memory size in System on Chip (SoC). SoC architecture normally occupies the majority of its area by memories. Due to increase in density of embedded memories, there is a need of self-testing mechanism in SoC design. Therefore, this research study focuses on this problem and introduces a smooth solution for self-testing. In the proposed memory test algorithm, the self-testing as well as self-repair mechanisms are incorporated. This scheme repairs the detected faults and is easily integrated with SoC design. Here, an attempt has been made to implement the memory built-in-self-repair (MBISR) architecture to test and repair the faults from the embedded memories. It is little, and it supports at-fast test without timing penalty during its operation. The proposed method is a better alternative in speed and low area overhead. Thus, it plays a significant role in yield improvement.

***doi**: 10.5829/ije.2017.30.06c.03*

1. INTRODUCTION

As the demand for embedded products increases day by day testing of embedded memories becomes a critical issue. Because embedded products are implemented on System on Chip (SoC) where memories reside in a largest part of chip area. Memories are more prone to failures than actual logic. Thus, deliberate more defects with complex defects type detection. To deal with such complex circuits dense with RAM (Random access memories), traditional test solutions do not provide complete solutions. Those methods fail to provide desired test frequency and test pattern length for large RAM. The methodology used to test RAM traditionally is the use of state machine to provide the set of test patterns with respect to algorithms. This way the state machine cannot provide correct test patterns if the complexity increases in the algorithm. In this case, it becomes very large and extremely slow to provide test patterns on each clock event and therefore fail to support the real speed to test RAM in SoC. The automatic test equipment (ATE) is also used to test memories conventionally, but this method increases complexity and cost of integrated circuit. As per today's deep submicron stage, testing embedded RAM in SoC becomes important to save chip yield [1-3].The percentage in an area of embedded RAM in SoC increases and memory occupying majority of area in SoC. To solve this problem, many researchers have provided their solutions on RAM testing. They provide their own methods/algorithms to improve the testability of embedded RAM [4]. These methods mostly used built-in- self-test (BIST) mechanisms, which provide excellent solutions to the issues. Thus, researchers have devoted themselves to develop BIST scheme for locating faults in embedded memories of SoC. The method of programmable BIST also introduces in literature [4, 5] as one of the solutions.

The traditional approach for programmable BIST for memories fault diagnosis, the microcode was used. The chosen algorithm for test stored in dedicated memory elements RAM, as an instruction code. Every time as test algorithm is changed the corresponding code word is being mapped and new BIST is achieved. This approach is indicated in research for process diagnosis and monitoring [5, 6]. Due to RAM based storages these methodologies suffers from several drawbacks, like area

*Corresponding Author's Email: *altaface1@gmail.com* (M. Altaf Ahmed)

Please cite this article as: M. Altaf Ahmed, D. Elizabath Rani, S. A. Sattar, Embedded Memory Test Strategies and Repair, International Journal of Engineering (IJE), TRANSACTIONS C: Aspects Vol. 30, No. 6, (June 2017) 839-845

overhead and low testability of BIST circuitry and low flexibility with supported test algorithms. Therefore, there is a need of sophisticated algorithm which works on these issues and provides a better solution. In this research study, we proposed an algorithm for testing the memories for fault and the mechanism to repair the diagnosed faults. This algorithm is named as Sift to detect various faults such as Stuck-at faults (SAFs), Address-Decoder faults (ADFs), Transition faults (TFs) and Coupling faults (CFs). Based on this algorithm the design is implemented to detect and repair the faults with a reasonable area over head as compared with the recent work. The rest of the paper is organized as, Section 2 explains about related work discussion. Section 3 consists of methodology and implementation. Section 4 provides results obtained and comparison to existing work. Section 5 describes conclusion and section 6 is references.

2. RELATED WORK

Since last three decade memory testing has been a research topic of interest. Many researchers have implemented memory BIST as well as Memory BISR designs. The MBIST architecture for covering different faults in the embedded memories are presented [2, 5, 7, 8].

The transparent based programmable MBIST for off line and on line testing has been presented and the area overhead is proposed [5]. The linear feedback shift register (LFSR) based testing for MBIST for low power is proposed [7]. Author has tried to minimize the dynamic power by decreasing the switching activity. A survey has been conducted in fault testing of memories and has discussed the area overhead in various approaches [8]. A memory BIST controller for detecting types of faults using March algorithm is proposed [2]. For industrial ASIC design, an approach is introduced to test the embedded RAMs. The above discussed work covers the diagnosis of memory for finding different types of faults in the memory.

The detected faults can be repaired, as mentioned in literature [9] by the spare memory approach named BRAINS. The survey of embedded memory test strategies and fault repairs are covered [8]. The area over head in SoC design is proposed [10] and is 4.1% in 8Kx64 memory size. The repair method is introduced with the additional area over head with redundancy architecture for DRAM as 1.3% [11].

Apart from this, many authors have proposed various fault detecting methods such as stuck-at fault, address decoding fault, transition fault and very few of them have covered the coupling faults. The repair methods are also described in few papers for low area over head. To continue this work we proposed a smart test solution to detect these faults in the memory with repairable feature and high coverage of fault. The proposed method covers the features and exceptional solutions for diagnosis fault in embedded memories and repairable strategies at high frequency with excellent fault coverage. The area over head is just 2.3% and is inversely proportional to the size of the selected Virtex target device.

3. METHODOLOGIES

3. 1. Built-in-Self-Test Principle The basic principle of BIST is illustrated in Figure 1. The BIST principle is, to design the circuit so that the circuit can automatically test itself for faults. This is actually the additional functionality incorporate into the design of the circuit to provide the self-testing features. It can also termed as self-verification at the time. It is capable to test the embedded memories without any external stimuli [12].

Thus, testing integrated circuit (IC) can be done by additional logic design integrated within the same IC. This special design is able to send some test pattern to test the design itself and compute the faults if presented. As shown, in principle, the BIST consists of four functional entities. First, the test pattern generator (TPG), second the BIST Controller, third the circuit under test (CUT) and fourth, the output response analyzer (ORA). The TPG is producing a chain of patterns for testing the CUT. The OR analyzer indicates the Fail/Pass for the design under test. The BIST controller performs the operation of testing the design and, it provides the BIST done signal after the test completion.

A set of stimuli is applied to the CUT and output compares with expected/reference output to test the good or bad circuitry. If comparison is passed it will go to start and a new test will be executed. If the test compared is failed it displays faulty circuitry and indicates the test is done. The flow of test strategy is shown in Figure 2. The input stimuli are applied to the circuit, through the test benches. The test vectors, therefore, are written on the test bench and will test the circuit by comparing the output.

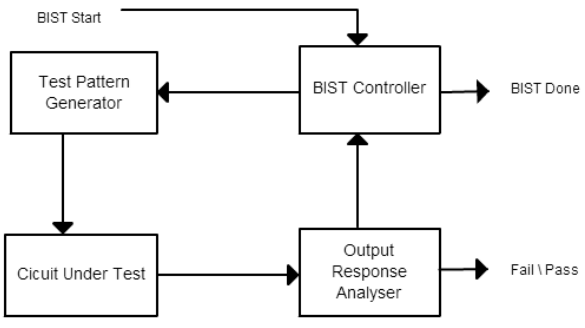

Figure 1. Basic BIST Principle

3. 2. Proposed Algorithm As discussed in the introduction part, the majority portion of silicon area in SoC is dominant by on-chip memories. The integration of high density memories on a single silicon chip might be the cause of many faults such as Stuck-at fault (SAFs), Transition fault, Coupling fault, Address Decoding faults etc. Let us introduce an approach to catch the faults which were appeared while dealt with the high capacity memories. The proposed algorithm here tests the embedded memory in SoC and named as Sift algorithm and mainly consists of several Sift elements, separated by semicolon. This algorithm is similar to the principle of MARCH based algorithm [13]. Sift algorithm is shown below with W and R notations which are known as read and write operations respectively.

{ ↕(W0) ; ↑(R0,W1); ↓(R1,W1,R1); ↑(R1,W0,R0);
↕R0 }

Sift Memory BIST Algorithm

This algorithm consists of ten (10N) Sift elements and five (5) Sift steps to accomplish the test. The up-arrow stands for ascending order of address sequence for writing and reading operation in each step. Similarly, the down arrow indicates the descending order of address sequence to perform writing reading operation in each step. These read write operations in front of the parenthesis are to be applied to each address one after the other sequentially. All the operations have to be performed before a jump to the next address location in the sequence of operations as per the algorithm steps. A fault can be caught in read operations only. The notations in the algorithm are indicated as follows.

- ↕: Indicates writing into and reading from the memory from address 0 to max or max to 0, order is not significant.
- ↑: Indicates writing / reading can be done from address 0 to max.
- ↓: Indicates writing / reading can be done from address max to 0.
- W0: Write all 0s to the address location.
- W1: Write all 1s to the address location.
- R0: Read 0s from the address location.
- R1: Read 1s from the address location.

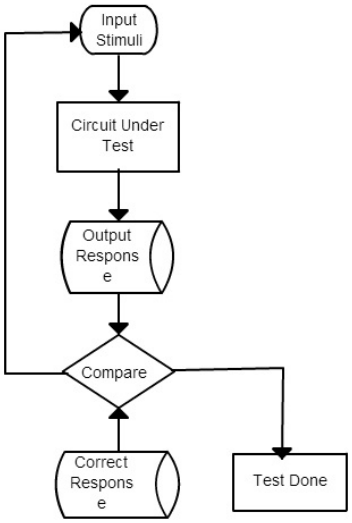

Figure 2. Test Flow

3. 3. Memory BIST Architecture Sift algorithm is implemented for fault diagnosis and repair of the detected faults. Memory BIST architecture for proposed method is indicated in Figure 3. It consists of main memory array (SRAM) on which the test is going conduct, Memory BIST Controller, reference test pattern storage block to detect and repair the defects presented in the embedded memory blocks [2]. The MBIST controller with these blocks performs on-chip self-testing of SRAMs.

The address Decoders are used for testing of selected addresses of memory. It uses internally counter or LFSR to produce the random addresses to test. MUX is used to select the data for comparing from main memory and storage pattern and it select the mode of MBIST. Comparator block compares the data out with expected output and results in the test pass or fail. MBIST controller is a state machine uses the separate state for all individual steps of the algorithm and performs read-write operations on memory indicate in Figure 4.

This is the main self-testing block which detects the error in main memory and intimates about the address at which error occurred. Then, to repair the detected fault the additional logic is introduced which selects the original data from storage and ignores the address location at which error is detected. The repair mechanism is illustrated in Figure 5 and it is discussed in subsequence subsection.

3. 3. 1. Memory BIST Controller State Machine Implementation Memory BIST Controller block is implemented using state machine consisting of the states equal to the operational steps in algorithm and

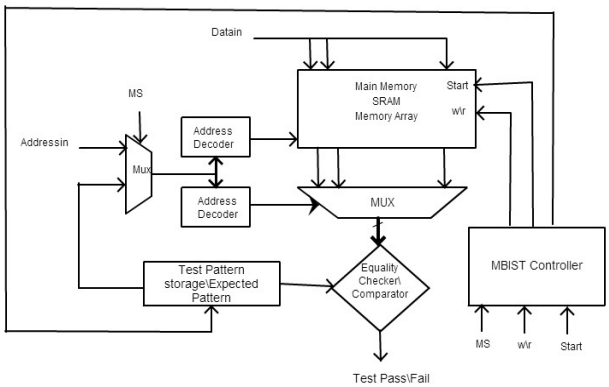

Figure 3. Memory Built in Self-Test Architecture

two additional states Idle for start and status to display results. The states of the state machine are as follows.

- S0: Idle: To start the memory BIST operation.
- S1: W0: When start signal asserts state machine jump to W0 state and perform write operations in ascending or descending (order is not significant) get filled with all 0s at all locations.
- S2: R0W1: This state performs read operation and then writes operation at the same address location then changes the address and performs the same for whole memory in ascending order of address sequence.
- S3: R1W1R1: This state performs read first then write and again read operation in descending order of address sequence and jumps to next state if the 0th address is reached.
- S4: R1W0R0: It is similar to previous state but writes 0s in place of 1s.
- S5: R0: This state reads 0s in any address order sequence.
- S6: Status: The state machine jump to this state from any state where read operations are carried out if a fault occurs and it displays result about fail memory Id and address location of the fault occurrence.

3. 3. 2. Built in Self Repair Architecture

Memory Built in Self Repair Architecture shown in Figure 5 consists of memory BIST controller which works according to algorithm and built in self-repair block. If fault detects during read operation a multiplexer at the Data out will select only the correct output that is from the spare memory location.

Therefore, the data flow is from memory under test if no faults detected and from spare memory if the failure occurs. The procedural flow chart of MBISR is shown in Figure 6. The performance of the BISR depends upon the size of spare/redundant array. The increase of faulty memory locations may tolerate but increases the area and complexity.

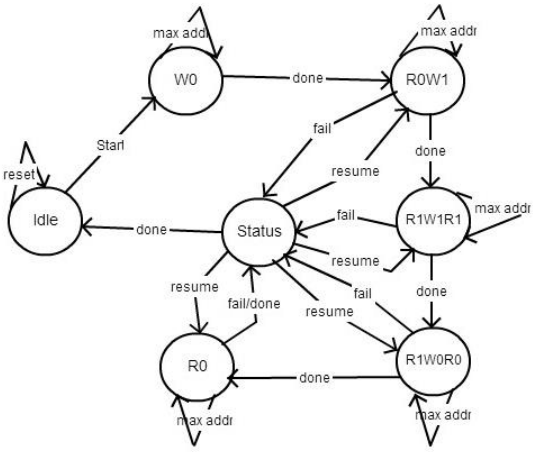

Figure 4. Memory BIST State Machine

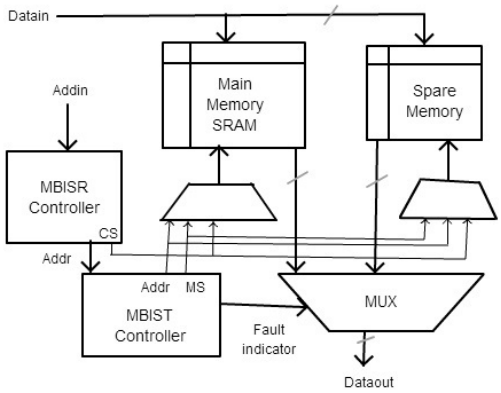

Figure 5. Built in Self Repair (BISR) Architecture

To avoid such circumstances of more faulty locations, complexity and area overhead, the MBISR circuitry must follow the shared mechanism for multiple memories. In this method, the MBISR will share to multiple RAMs in parallel and serial shared BISR [14, 15] which is considered to be the solution to this problem. The shared BIST scheme is therefore mentioned in Figure 7.

It consists of multiple RAM blocks and BISR circuitry which is shared by all the block of memory. This is a generalized form of SoC where embedded memory blocks are placed. The BISR circuitry denoted by g1 is shared with embedded memory blocks under test M1, M3, M4 and M5. The BISR circuitry which is identified as g2 is shared into memory blocks M2, M6, M7 and M8 under test.

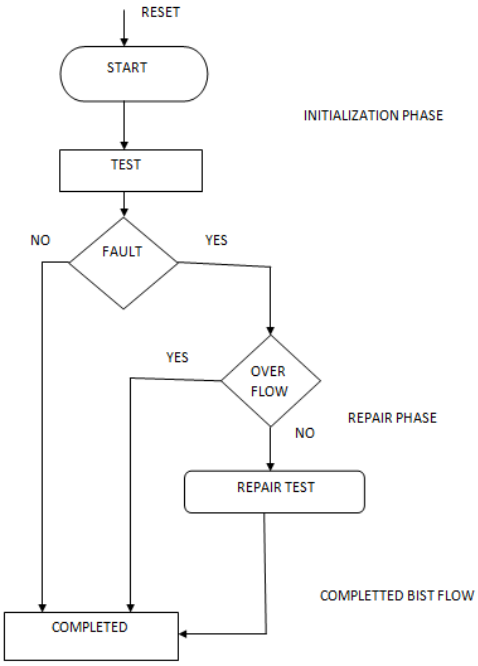

Figure 6. MBISR Chart

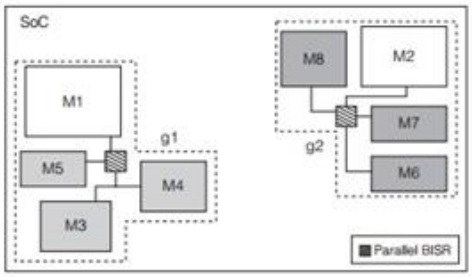

Figure 7. BISR for Multiple RAM Blocks

4. RESULTS AND COMPARISON

The architecture is implemented in Verilog Hard Ware Description language (HDL). The Simulation and synthesis carried out using Modalism and Xilinx ISE 14.1 tool respectively [3, 16]. To simulate the architecture, test cases are written to verify the functional correctness. The multiplexer is used to perform more than one operation in the same location of memory in several states of the state machine. The simulation result for writing and reading into memory is shown in Figure 8.

The Sift algorithm consists of several Sift elements as mentioned in the algorithm the state machine implemented consisting of seven states. In the simulation, two cases are considered for memory M1 and memory M2. The M1 is failed due to the defect occurred at memory address location 2. Once the fault is detected the state machine jumps to status state and displays the information about memory Id, fail address location and pass/fail status. In the second case for memory Id M2, it is passed, because no fault occurred during the self-test procedure. Synthesis results are tabulated in the Table 1.

The architecture is implemented on Virtex 7 FPGA with Target Device xc7vx330t-3-ffg1157 [16-18] selected. The total area occupied is 268940 kilobytes. This is just 2.13 percent area overhead of total chip area. The total area in slice LUTs are 160 (uses 69-LE) and the time period is 2.10ns. The maximum working frequency of hardware is 496.43 MHz.

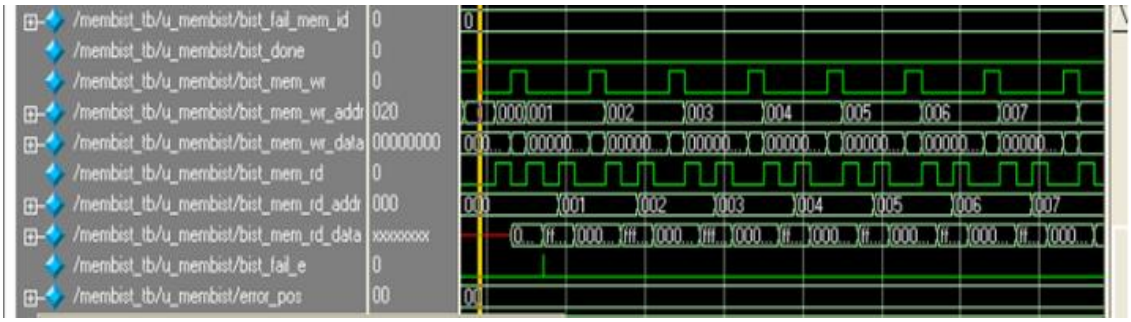

Figure 8. Simulation Result for memory writing and reading

TABLE 1. Synthesis Results

Design	Area Slice LUTs	Time	Max .Frequency (MHz)
MBIST Controller (Proposed)	69-LE (160 LUTs)	2.10ns	496.43
MBIST Controller for MARCH C+	71-LE	4.068ns	245.821
PMBIST (MARCH SAM)	77-LE	----	80.27
Hybrid P-MBIST	81-LE	----	82.17

The proposed algorithm is therefore compared with other existing algorithms. The results show that the proposed algorithm is better in terms of fault diagnosis and self-testing. The comparison of various March algorithm and targeted faults are indicated in tabular format in Table 2. This algorithm targets variety of faults occurred in the memory.

TABLE 2. Comparison with March Algorithms

Sr. No.	Algorithm	No.of Elements	No.of steps	Test sequence	Target Faults
1	March A	15N	5	*	SAFs, ADFs, TFs,
2	March X	6N	4	**	CFs
3	Mats +	5N	3	***	SAFs, ADFs
4	March C	11N	7	****	SAFs, ADFs, TFs some CFs
5	Proposed Sift	10N	5	*****	SAFs, ADFs, TFs, CFs

* {↕W0, ↑(R0,W1,W0, W1), ↑(R1,W0,W1), ↓((R1,W0,W1, W0), ↓(R0,W1, W0)}

** {↕W0, ↑(R0,W1), ↓(R1,W0), ↕R0

*** {↕W0, ↑(R0, W1), ↓(R1,W0)}

**** {↕W0, ↑(R0,W1), ↑(R1,W0), ↕R0, ↓(R0,W1), ↓(R1,W0), ↕R0 }

***** { ↕W0, ↑(R0,W1), ↓(R1,W1,R1), ↑(R1,W0,R0), ↕R0}

TABLE 3. MBIST Comparison

Sr. No.	Method	Supported Algorithm	Target Faults	Fault Coverage %	Repairable
1	Shi-Yu Hung [14] MBIST	March	SAFs	Low	No
2	Sanghun Park [4] MBIST	Extended March -C	SAF, ADFs	Low	No
3	MBIST Video Decoder [19]	March 17N	SAFs, ADFs, TFs	Low	No
4	MBIST Dongky u Youn, [13]	March C	SAFs, ADFs, TFs	Low	No
5	Proposed MBIST	Sift	SAFs, ADFs,TFs, CFs	High	Yes

The comparison in Table 3 describes the target faults coverage, and the repairable features.

It can be concluded that the higher and better fault coverage are obtained from the proposed algorithm as compared with other approaches. The fault coverage is high with the proposed method because it targets almost all possible faults occurred in the memory. The additional feature of repair for detected faults is also embedded. The comparison with other mentioned methods are tabulated.

5. CONCLUSION

The proposed method MBISR provides fault diagnosis and supports repair schemes for embedded memory blocks. The architecture is implemented for easy integration in SoC with little area overhead and high repairable rate. The experimental results show that the MBISR works on max. frequency of 496.43 MHz with only 160 slices LUT and 2.10ns time period. The method successfully detects maximum faults occurred and repairs them in the memory. Thus, it helps in improving the overall yield of the chip with very low area overhead. It works at high speed with only 2.13 percentage of total chip area overhead. As a result, we conclude that the proposed architecture can lead to a significantly faster and has ability to test and repair the embedded memory in SoC. It plays an essential role in improving the total yield of product.

6. REFERENCES

1. Wu, T.-H., Chen, P.-Y., Lee, M., Lin, B.-Y., Wu, C.-W., Tien, C.-H., Lin, H.-C., Chen, H., Peng, C.-N. and Wang, M.-J., "A memory yield improvement scheme combining built-in self-repair and error correction codes", in Test Conference (ITC), IEEE International,. (2012), 1-9.
2. Ahmed, M.A., Rani, D.E. and Sattar, S.A., "Fpga based high speed memory bist controller for embedded applications", ***Indian Journal of Science and Technology***, Vol. 8, No. 33, (2015).
3. Aljumah, A. and Ahmed, M.A., "Amba based advanced dma controller for soc", ***International Journal of Advanced Computer Science and Applications***, Vol. 7, No. 3, (2016), 188-193.
4. Park, S., Lee, K., Im, C., Kwak, N., Kim, K. and Choi, Y., "Designing built-in self-test circuits for embedded memories test", in ASICs,. AP-ASIC. Proceedings of the Second IEEE Asia Pacific Conference on, (2000), 315-318.
5. Boutobza, S., Nicolaidis, M., Lamara, K.M. and Costa, A., "A transparent based programmable memory bist", in Test Symposium,. ETS'06. Eleventh IEEE European, (2006), 89-96.
6. Schanstra, I., Lukita, D., Van de Goor, A.J., Veelenturf, K. and van Wijnen, P.J., "Semiconductor manufacturing process monitoring using built-in self-test for embedded memories", in Test Conference, Proceedings., International, (1998), 872-881.
7. Krishna, K.M. and Sailaja, M., "Low power memory built in self test address generator using clock controlled linear feedback shift registers", ***Journal of Electronic Testing***, Vol. 30, No. 1, (2014), 77-85.
8. Acharya, G.P. and Rani, M.A., "Survey of test strategies for system-on chip and it's embedded memories", in Intelligent Computational Systems (RAICS), IEEE Recent Advances in, (2013), 199-204.
9. Lee, M., Denq, L.-M. and Wu, C.-W., "A memory built-in self-repair scheme based on configurable spares", ***IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems***, Vol. 30, No. 6, (2011), 919-929.
10. Su, C.-L., Huang, R.-F., Wu, C.-W., Luo, K.-L. and Wu, W.-C., "A built-in self-diagnosis and repair design with fail pattern identification for memories", ***IEEE Transactions on Very Large Scale Integration (VLSI) Systems***, Vol. 19, No. 12, (2011), 2184-2194.
11. Lin, B.-Y., Chiang, W.-T., Wu, C.-W., Lee, M., Lin, H.-C., Peng, C.-N. and Wang, M.-J., "Redundancy architectures for channel-based 3d dram yield improvement", in Test Conference (ITC), IEEE International, (2014), 1-7.
12. Stroud, C.E., "A designer's guide to built-in self-test, Springer Science & Business Media, Vol. 19, (2002).
13. Youn, D., Kim, T. and Park, S., "A microcode-based memory bist implementing modified march algorithm", in Test Symposium,. Proceedings. 10th Asian, IEEE. (2001), 391-395.
14. Huang, S.-Y. and Kwai, D.-M., "A high-speed built-in-self-test design for drams", in VLSI Technology, Systems, and Applications,. International Symposium on, IEEE., (1999), 50-53.
15. Tseng, T.-W., Li, J.-F. and Hou, C.-S., "A built-in method to repair soc rams in parallel", ***IEEE Design & Test of Computers***, Vol. 27, No. 6, (2010), 46-57.
16. Aljumah, A. and Ahmed, M.A., "Design of high speed data transfer direct memory access controller for system on chip based embedded products", ***Journal of Applied Sciences***, Vol. 15, No. 3, (2015), 576-582
17. Zivarian, H., Soleimani, M. and Mohammadi, M.D., "Field programmable gate array–based implementation of an improved algorithm for objects distance measurement (technical note)", ***International Journal of Engineering-Transactions A: Basics***, Vol. 30, No. 1, (2016), 57-64.
18. Mandal, A. and Mishra, R., "Design and implementation of digital demodulator for frequency modulated cw radar", ***IJE Trans. A: Basics***, Vol. 27, No. 10, (2014), 1581-1590.

19. Hou, L., Wu, W. and Zhu, J., "Mbist design and implementation of a h. 264/avc video decoder chip", in Signal Processing Systems (ICSPS), 2nd International Conference on, IEEE. Vol. 1, (2010), V1-87-V81-90.

Embedded Memory Test Strategies and Repair

M. Altaf Ahmed[a], D. Elizabath Rani[a], S. A. Sattar[b]

[a] *Department of ECE, GITAM Institute of Technology, GITAM University, Visakhapatnam, India*
[b] *Department of ECE, Royal Institute of Technology & Science, A. P. India*

P A P E R I N F O

Paper history:
Received 13 July 2016
Received in revised form 24 April 2017
Accepted 24 April 2017

Keywords:
Embedded Memory
Self-testing
Memory Built-in self-repair
System on Chip
Memory Test Algorithm
Yield Improvement

چکیده

درخواست برای خودآزمونی با اندازه حافظه در سیستم بر چیپ(SoC) فزونی یافته است. معماریSoC معمولا بخش عمده سطح را با حافظه پر می کند. با توجه به افزایش چگالی حافظه های جاسازی شده، در طراحی SoC نیاز به سازوکار خود آزمونی است بنابراین، این تحقیق بر این مساله متمرکز بوده ویک حل هموار برای خود آزمونی معرفی می کند. در الگوریتم آزمون حافظه پیشنهادی علاوه بر خود آزمونی، سازوکار بهسازی نیز شرکت دارد. این شما خطا های پیدا شده را اصلاح کرده و به سادگی با طرح SoC یکپارچه می شود. در اینجا کوشش شده است تا از روی حافظه های جا سازی شده، معماری حافظه built-in-self-repair (MBISR) را برای آزمایش و اصلاح خطا ها پیاده سازی کند. این کار سریع و بدون جریمه زمانی طی عملیات مربوط صورت می گیرد. روش ارائه شده شیوه بهتری از نظر سرعت و با سطح کم می باشد و بنا براین نقش قابل توجهی در بهبود نتیجه دارد.

***doi**: 10.5829/ije.2017.30.06c.03*