International Journal of Information and Computation Technology.
ISSN 0974-2239 Volume 3, Number 5 (2013), pp. 361-370
© International Research Publications House
http://www. irphouse.com /ijict.htm

Built-in Self-repair Mechanism for Embedded Memories using Totally Self-checking Logic

[1]G. Prasad Acharya and [2]M. Asha Rani

[1]Sreenidhi Institute of Science and Technology, Hyderabad, A.P., India.
[2]JNTU College of Engineering, Hyderabad, A. P., India.

Abstract

Today's deep submicron technologies allow the integration of multiple memories on a single chip. Embedded memories are one of the most universal cores, which occupy around 90% area in system-on-chip (SoC) architecture due to the demand for high data storage. So, the embedded memory test process has become an essential part of the SoC development phase. Some of the March algorithms including recently developed March SS algorithms though provide architectural flexibility to run different algorithms on the similar hardware architecture. This flexibility is a trade-off with logic overhead, number of March operations and computational delay. The MBSIR methodology presented in this paper is based on the totally self-checking logic using Berger code. This architecture supports the fault diagnosis as well as on-fly self-repair mechanism. The experimental results show that the complexity of the replacement of the faulty core and its impact on performance degradation may be traded-off by a marginal increase in test-time and logic overhead.

Keywords: Self-repair, Redundant Memory Array, Berger Code, MBISR Controller, Self-Checking Logic.

1. Introduction
Today's semiconductor memories ranging from gigabytes to terabytes of capacity consist of millions to billions of transistors, diodes and other components such as capacitors and resistors, together with interconnections, within a very small Silicon area. The technology shrinking had given rise to the introduction of parasitic

capacitances, resistive opens and shorts causing more leakage power and adverse effects in its performance. The manufacturing of such circuits is a complicated and time-consuming process and defects in them are inevitable. The System-on-Chips which were earlier dominated by functional modules are now being dominated by the memory chips because of the demand for high storage requirement. In addition to the above mentioned parasitic components, transistor short channel effect, cross talk effects, impact of process variation have evolved new fault models for embedded memories. The commonly applied functional fault models were the Stuck-At Fault (SAF), the Address decoder Fault (AF), the Coupling Fault (CF), and the Neighborhood Pattern Sensitive Fault (NPSF) model. These fault models provides the platform for the developments of fault diagnosis algorithms to improve test time coverage, test time and repair efficiency, thereby improving the yield.

The next section gives a brief outline about the Built-in-Self-Test methodology used for testing of any circuitry. The Memory Built-in-Self Repair technique and its architecture is presented in section III.

2. Built-in Self Test Architecture

Built-In Self-Test (BIST) mechanism used for testing of manufactured ICs has the capability of testing the circuit itself by incorporating the Automatic Test Pattern Generator (ATPG) and Output Response Analyzer (ORA) within a marginally increased logic overhead and Silicon area. The various memory BIST (MBIST) techniques [5,7,8] provide the most cost-effective and widely used solutions for embedded memory testing because of the following reasons: (1) Requires no external Automatic Test Equipment (ATE); (2) Reduced design modifications and efforts and increased flexibility; (3) Tests can be run At-speed to reduce the test-time and cost; (4) Improved controllability and observability because of increased number of test access points on the chip (5) On-chip test pattern generation and output response analysis; (6) Test may be carried out both on-line and off-line; (7) Adaptability to technology. The challenges for memory BIST include the generation of complex test pattern generation algorithms which can target the various types of faults in embedded memories and the delays for replacing the faulty memory chips. In case of memory chip in which the occurrence of the faults is very rare, the memory built-in-self-repair algorithm [4, 6] will be more cost-effective and less performance degrading especially for real-time embedded systems as compared to the replacement of faulty memory ICs physically from the system-on-chip. The recently developed march SS algorithm [9] provides the Built-In Self Test and repair mechanism for embedded memories in SOC and also facilitates the architectural flexibility to run different March algorithms on the similar hardware architecture. This algorithm has the drawbacks of requiring large number of complex March operations causing more computational delay and area overhead.

3. Proposed MBISR Architecture

Built-in-Self-Repair architecture provides a better and cost-effective solution for fault tolerance of embedded memories on the system-on-chip with the help of redundant modules/chips. The BIST architecture can be modified to facilitate the self-repair mechanism by incorporating the redundant memory array as shown in figure shown in Figure 1. The proposed memory BISR architecture consists of MBISR controller, Redundant Memory Array, Memory Under Test (MUT), an Output Response Analyzer and Self-checking Logic which detects the presence or absence of a fault in memory chip.

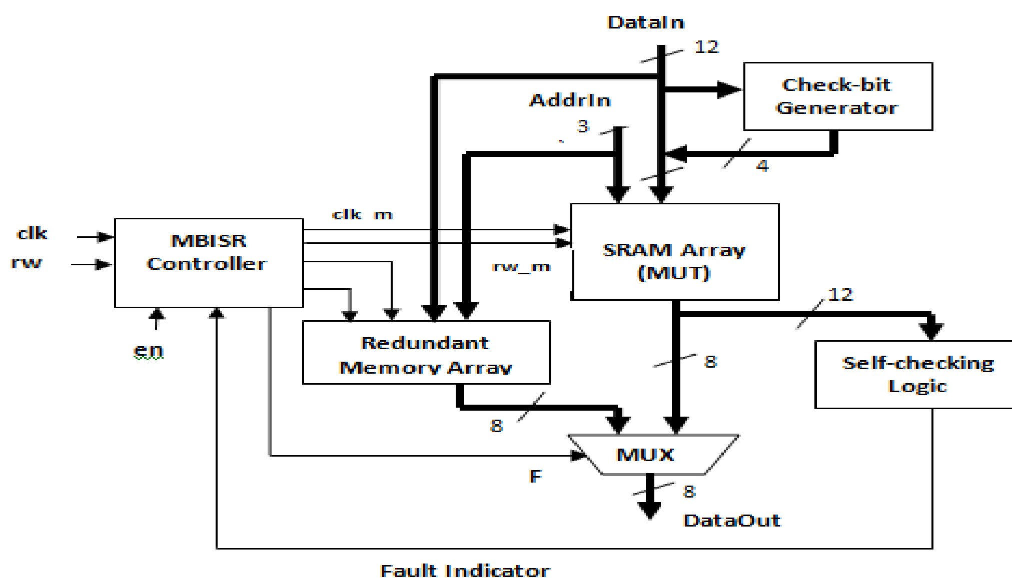


Figure 1: Memory BISR architecture.

The redundant memory array is used to store and retrieve the data in case a memory location is detected as a faulty location. A combination of DataIn and the k-check bits produced by check-bit generator using the Berger Code logic, which represent the number of 0's in the data word is written to the MUT. The DataOut after performing read operation is fed to a totally self-checking logic to detect the presence of faults, if any in MUT.

The self-checking checker notifies the presence or absence of faults in the MUT to the MBISR controller, which then instructs the Redundant Memory Array to fetch the correct data in case a faulty location is detected. The 2x1 MUX then presents the correct data based on the fault indicator flag. The functional blocks of the above proposed architecture is described as follows:

3.1 Check-Bit Generator

The number of check bits, k for a data/information sequence of length I bits is evaluated such that it satisfies the inequality $I=2^k-1$. The check bits generated by the check-bit generator shown in Figure 1 indicate the number of 0's being present in the information bits. A zero's counter may be used to keep the record of number of 0's in the information bits. A codeword is generated by concatenating the data bits followed by check bits, thereby forming the codeword separable.

3.2 Redundant Memory Array

The Redundancy Logic comprises of placing a Redundant Memory Array (RMA) in parallel with the MUT. A threshold is defined as the maximum number of faulty locations that can be tolerated in a reusable MUT and is used to define the size of RMA. Considering the increased complexity and chip area, only 5-10% of the overall memory capacity may be incorporated in the redundant Array. The memory location to store data (in case a faulty location is detected in 2^NxM MUT) in the Redundant Array may be computed as $A_{ri} = A_i \bmod N$, where N is the number of address lines (=4), M (=12) is the word size of the memory, A_i is the faulty i^{th} address location in MUT and A_{ri} is the corresponding i^{th} address location in the RMA. A copy of the DataIn written to the MUT at a location A_i specified by the AddrIn is also made available in the corresponding location A_{ri} of Redundant Array. The RMA is used to retrieve the data in case a faulty word is detected during testing. Once any faulty location is detected, the faulty address is captured by MBISR controller and then it facilitates accepting the data from the redundant memory. The fault indicator output (F) enables 2x1 MUX to select the data to be read-out either from the MUT or from the Redundant Memory Array.

3.3 Self-Checking using Berger Code

A general structure for a totally self checking checker [1-3] using a Berger code is shown in Figure 2. A Berger code is said to be of maximal length if $I=2^k-1$, otherwise of non-maximal length. The combinational circuit C1 produces the complement of the k-check bits (=4 bits). For a maximal-length Berger code, it gives the count of the number of 1's in the information bits, whereas the check-bits for a non-maximal length Berger code is the binary representation of 2^k-N_0, where N_0 is the count of zeros in the information bits.

The two-rail checker implements k-out-of 2k (K=4) checker logic by accepting k-check bits generated by check-bit generator and k-check bits corresponding to the data part stored in MUT, produced by combinational circuit C1. Thus, the two sets of complemented check-bits are being presented to the two-rail checker. The two rail-checker produces two outputs f and g, that are complement to each other in absence of any faults in MUT, otherwise it produces the same outputs. Logic-'1' at the output of XNOR-gate indicates the presence of fault in circuit.

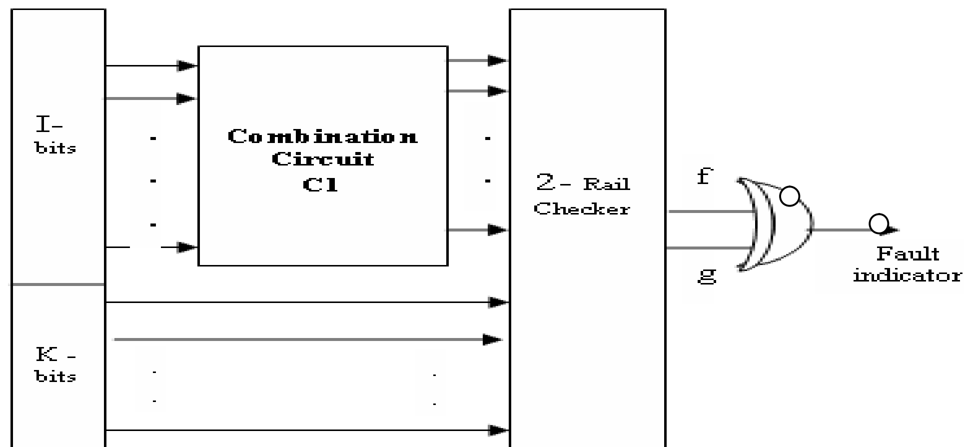


Figure 2: Totally Self-checking checker.

3.4 MBISR CONTROLLER

The MBISR controller produces the control signals required for proper functioning of MBISR logic. The operation of MBISR controller is explained with the help of flow chart shown in Figure 3. The MBIST controller includes the following control signals:

i. Control signals for MUT

MBIST controller enables the MUT chip by asserting chip enable signal. It then activates the memory-write and memory read operation on positive edge of "clk" by asserting and de-asserting "rw" signal respectively.

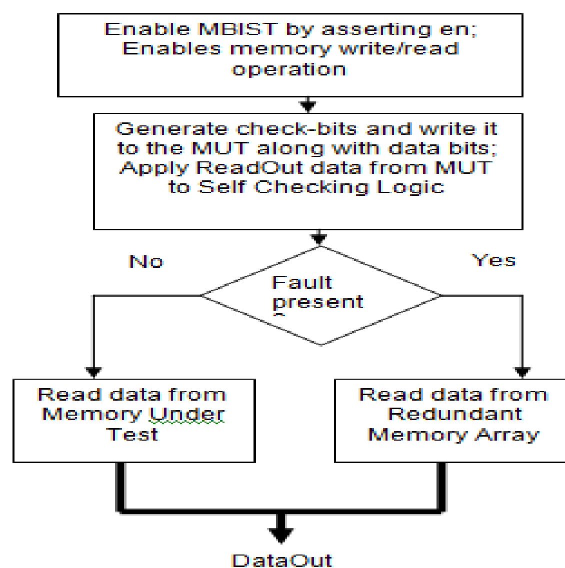


Figure 3: Flow chart of MBIST operation.

ii. Control signals for Redundant Memory Array

MBIST controller detects the presence of any fault in MUT by observing fault_indicator (Logic-'1' value). In case a fault is detected in MUT, the MBIST controller enables the Redundant Memory Array and then it allows the read operation on the Redundant Memory Array by de-asserting "rw_red" signal so that the correct data may be driven at the output of MUX thereby providing the built-in-self-repair feature.

iii. MUX Selection input (F)

This signal enables the correct data to be read from the corresponding location of Redundant Memory Array in case a faulty location is detected in the MUT.

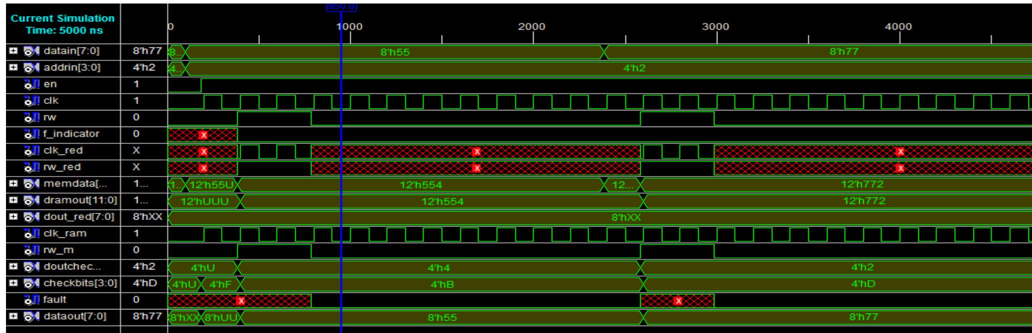
4. Result Analysis

The source code for the Memory BISR scheme proposed in this paper has been written in VHDL and then simulated in Xilinx environment. The practically simulated waveforms have been analyzed and verified with the theoretical observations. To illustrate the proposed MBISR mechanism, a 24x12 MUT was taken as a design example. In the 12-bit word being accessed from each of 16-locations (4-address lines), the first 8-bits represent the information and the remaining 4-bits indicate the check bits generated using Berger code logic.

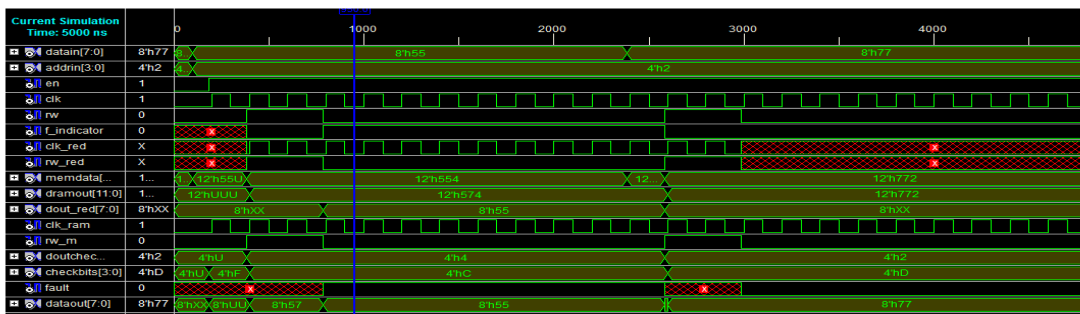
The MBISR controller generates the appropriate control signals for MUT and RMA. The simulated waveform in Figure 4(a) shows that the control signals "clk_red" and "rw_red" for the RMA are generated if and only if any fault is detected by the self-checking checker circuit. It then enables the write/read operations to/from the redundant array and also notifies the presence of fault in the MUT.

Figure 4(b) presents the simulated waveforms of the proposed MBISR top module in absence of any faults. An 8-bit data along with 4-generated check bits (=554H) has been written in the address location 2H during the assert period of "rw" signal, and then read out when "rw" goes low. In absence of any faults, the RMA remains disabled and the MUX reads the data from MUT.

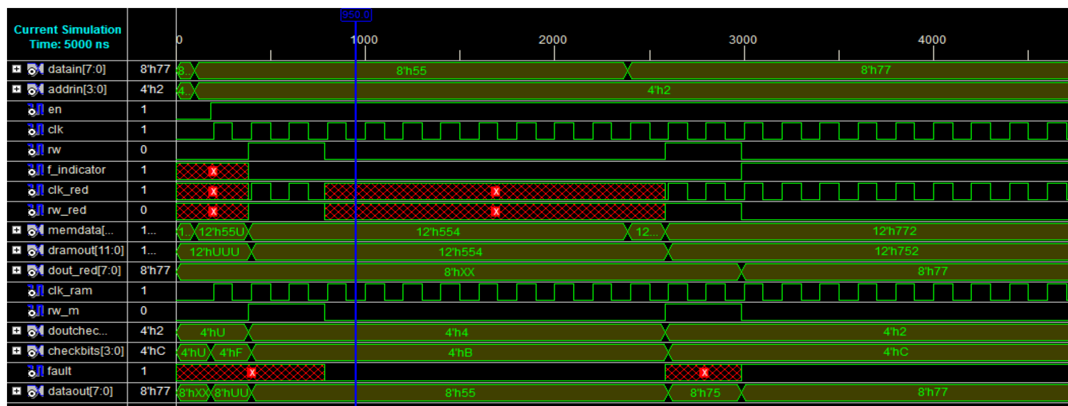
(a) Simulated waveforms of MBISR controller.



(b) Simulated waveforms in absence of any faults in MUT.



(c) Simulated waveforms detecting the presence of fault (S-A-1) in the MUT



(d) Simulated waveforms showing the presence of fault (S-A-0) in the MUT.

Figure 4 (a-d): Simulation results of Memory BISR Architecture.

The top level module was simulated by injecting a fault (S-A-1) at bit-position 5 of address location 2H. The simulated result shown in Figure 4(c) shows the presence of fault; it then enables the RMA for write operation to store the input data. The MUX then reads the correct data from RMA. The input data 77H writes logic-1 at bit position 5 and hence makes it ambiguous for the detection of S-A-1 fault at that bit-location. Similarly, Figure 4(d) shows the presence of S-A-0 fault at the same bit location.

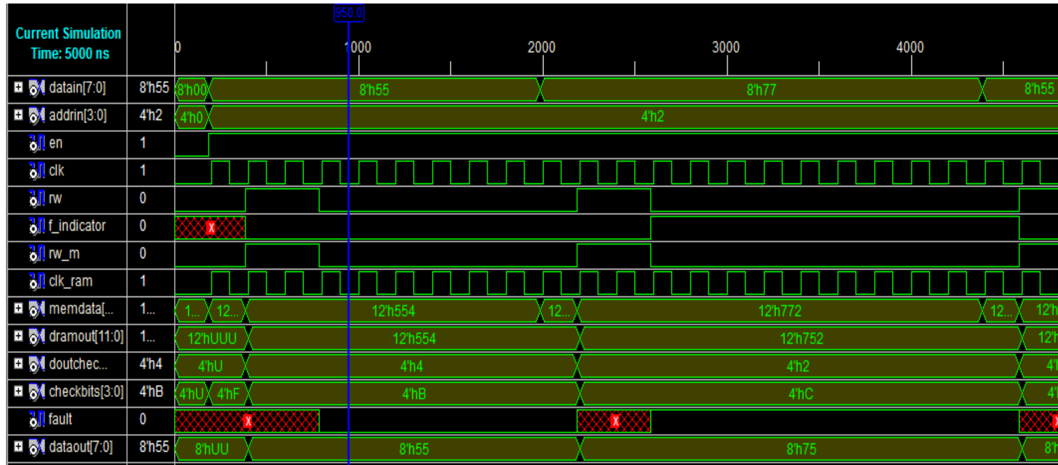


Figure 5: Simulation results of Memory BIST Architecture.

The BIST architecture for 16x12 SRAM is also implemented and simulated for comparison purpose. The MBIST architecture is capable of only indicating the presence of faults in the MUT, but not capable of retrieving the correct data in presence of faults (The simulated waveform shown in Figure 5).

The design is synthesized on XC3S400 device using Xilinx Synthesis Tools. Table I summarizes the device utilization of MBIST and MBISR architecture for different sizes of SRAMs. The FPGA implementation of MBISR architecture of 1Kx12 SRAM requires less than 2% of logic overhead as compared to that of MBIST architecture. As the size of memory increases from megabytes to terabytes, the logic overhead of MBISR architecture becomes negligible.

5. Conclusions

The self-repair capability of the embedded memory using Self-checking Logic is implemented and verified on the target Sparton 3 FPGA board. The self-repair mechanism certainly helps to minimize the delay for identifying and replacing the faulty memory core from the system-on-chip for a tolerable number of faults in it. The self-repair capability of embedded RAM chip has been verified by injecting the Stuck-at faults in the certain memory locations. The obtained synthesis results show that the on-fly self-repair capability of memory BIST technique can be introduced in MBISR technique with a negligible circuit overhead and complexity.

Table 1: Comparison of device utilization of 16x12 Single-port RAM with MBISR and MBIST capabilities.

	Single-Port RAM		# slices	# flip-flops	# 4-i/p LUTs	# IOs	# bonded IOBs	# Gclks	% of logic overhead in MBISR architecture	
	MUT	RMA							Slices	4-i/p LUTs
MBIST	16x12	NA*	**35**	8	**63**	24	24	3	-	-
	256x12	NA*	**153**	8	**299**	28	28	3	-	-
	1024x12	NA*	**543**	8	**1046**	30	30	3	-	-
MBISR	16x12	4x8	**45**	8	**82**	24	24	3	28.57	30.16
	256x12	4x8	**162**	8	**316**	28	28	3	5.88	5.69
	1024x12	4x8	**552**	8	**1063**	30	30	3	1.66	1.63

*NA- Not Available.

References

[1] M. Mrouf and A. D. Friedman, "Design of self-checking checkers for Berger Codes", Digest of Papers 8th Annual International Conference on Fault-Tolerant Computing, pp. 179-184, June, 1978.

[2] Cecilia Metra, Michele Favalli, Bruno Ricco, "Novel Berger Code Checker", International Workshop on Defect and Fault Tolerance in VLSI Systems, 1995, pp 287-295.

[3] Cristiana Bolchini, Fabio Salice and Donatella Sciuto "Designing Self-Checking FPGAs through Error Detection Codes", Proc. Of the 17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2002.

[4] Y. Zorian, S. Shoukourian, "Embedded Memory Test and Repair: Infrastructure IP for SOC Yield", IEEE Design & Test of Computers, Vol. 20, No. 3, May-June, 2002, pp.58-66.

[5] Balwinder Singh, Sukhleen Bindra Narang, Arun Khosla "Modeling and Simulation of Efficient march Algorithm for Memory Testing", Communications in Computer and information Science, Volume 95, 2010, pp 96-107.

[6] Dr. R. K. Sharma, Aditi Sood, "Modeling nd Simulation of Multi-operation Microcode based Built-in-Self Test for Memory Fault Detection and Repair", IEEE Annual Sympoism on VLSI, 2010, pp 381- 386.

[7] Said Hamdioui, Georgi Gaydadjiev and Ad J. van de Goor, "The State-of-art and Future Trends in Testing Embedded memories", Records of the 2004 International Workshop on Memory Technology, Design and Testing, IEEE

[8] Said Hamdioui, Georgi Gaydadjiev and Ad J. van de Goor, "Testing Static and Dynamic Faults in Random Access memories", In Proc of IEEE VLSI Test Symposium, pp 395-400, 2002.

[9] Said Hamdioui, Ad J. van de Goor and M. Rodgers, "March SS: A Test for all Static Simple SRAM Faults", Proc. of IEEE International Workshop on Memory Technology, Design and Testing, pp 95-100, France, 2002.