International Journal of
Scientific Engineering and Technology Research

ISSN 2319-8885
Vol.04,Issue.40,
October-2015,
Pages:8840-8847

www.ijsetr.com

Programmable BIST for Memory System

KUNSOTH SANTHOSH
[1]PG Scholar, Dept of ECE(VLSI), Gokaraju Rangaraju Institute of Engineering & Technology, JNTUH, Hyderabad, India.

Abstract: 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 selfchecking logic using Berger code. This architecture supports the fault diagnosis as well as on-fly self-repair mechanism. The BISR is designed flexible that it can provide four operation modes to SRAM users. Each fault address can be saved only once is the feature of the proposed BISR strategy. Besides, instead of adding spare words, rows, columns or blocks in the SRAMs, users can select normal words as redundancy.

Keywords: BISR, BIST, SRAM

I. 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. 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. To solve the problem, a new redundancy scheme is proposed in this paper. Some normal words in embedded memories can be selected as redundancy instead of adding spare words, spare rows, spare columns or spare blocks. Memory test is necessary before using redundancy to repair. Design for test (DFT) techniques proposed in 1970 improves the testability by including additional circuitry.

The DFT circuitry controlled through a BIST circuitry is more time-saving and efficient compared to that controlled by the external tester (ATE) [7]. However, memory BIST does not address the loss of parts due to manufacturing defects but only the screening aspects of the manufactured parts [8]. BISR techniques aim at testing embedded memories, saving the fault addresses and replacing them with redundancy. In [9], the authors proposed a new memory BISR strategy applying two serial redundancy analysis (RA) stages. [10] Presents an efficient repair algorithm for embedded memory with multiple redundancies and a BISR circuit using the proposed algorithm. All the previous BISR techniques can repair memories, but they didn't tell us how to avoid storing fault address more than once. This paper proposes an efficient BISR strategy which can store each fault address only once.

II. 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:

- Requires no external Automatic Test Equipment (ATE); (2) Reduced design modifications and efforts and increased flexibility;
- Tests can be run At-speed to reduce the test-time and cost;
- Improved controllability and observability because of increased number of test access points on the chip
- On-chip test pattern generation and output response analysis;
- Test may be carried out both on-line and off-line;
- 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

Copyright @ 2015 IJSETR. All rights reserved.

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. The test process is generally done by the external device called ATE (Automatic Test Equipment). The ATE applies a pre-generated set of inputs called as test vectors to the chip and compares the outputs to a predefined set of expected values. The disadvantages of using ATE for testing are

- Cannot perform At-Speed testing.
- Very expensive i.e. cost increases in proportion to the number of test vectors.
- One ATE machine can only test one chip at a time there by causing delays for systems that are to be manufactured on a large scale.

These disadvantages can be overcome by BIST. BIST requires a small amount of extra hardware to be inserted into the chip. The basic concept of BIST involves the design of test circuitry around a system that automatically tests the system by applying certain test stimulus and observing the corresponding system response. The definition of the BIST is given as the ability of logic to verify a failure-free status automatically, without the need for externally applied test stimuli and without the need for the logic to be part of a running system.

A. Basic BIST

The various components of BIST hardware are the test pattern generator (TPG), the test controller, circuit under test (CUT), input isolation circuitry and the output response analyzer (ORA).

Test Pattern Generator (TPG): Responsible for generating the test vectors according to the desired technique (i.e. depending upon the desired fault coverage and the specific faults to be tested for) for the CUT. Linear feedback shift register (LFSR) and pseudo random pattern generator (PRPG) are the most widely used TPGs.

Test Controller: It is responsible for controlling the other components to perform the self test. The test controller places the CUT in test mode and allows the TPG to drive the circuit's inputs directly. During the test sequence, the controller interacts with the ORA to ensure that the proper signals are being compared. The test controller asserts its single output signal to indicate that testing has completed, and that the ORA have determined whether the circuit is faulty or fault-free.

Output Response Analyzer (ORA): Responsible for validating the output responses i.e. the response of the system to the applied test vectors needs to be analyzed, Also, a decision is made about the system being faulty or fault-free. LFSR and multiple input signature register (MISR) are the most widely used ORAs.

Classification of Test Pattern: The fault coverage that we obtain for various fault models is a direct function of the test patterns produced by the TPG. There are several classes of test patterns. TPGs are sometimes classified according to the class of test patterns that they produce. The different classes of test patterns are briefly described below.

Deterministic Test Patterns: These test patterns are developed to detect specific faults and/or structural defects for a given CUT. The deterministic test vectors are stored in a ROM.

- **Algorithmic Test Patterns:** Algorithmic test patterns are specific to a given CUT to test for specific fault models. Because of the repetition and/or sequence associated with algorithmic test patterns, they are implemented in hardware using finite state machines (FSMs).
- **Exhaustive Test Patterns:** Every possible input combination for an N-input combinational logic is generated and the exhaustive test pattern set will consist of 2^Ntest vectors. This number could be really huge for large designs, causing the testing time to become significant. TPG could be implemented using an N-bit counter.
- **Pseudo-Exhaustive Test Patterns:** The large N-input combinational logic block is partitioned into smaller combinational logic sub-circuits. Each of the M-input sub-circuits (M<N) is then exhaustively tested by the applying allthe possible 2^K input vectors. TPG could be implemented using counters, Linear Feedback Shift Registers (LFSRs).
- **Random Test Patterns:** A truly random test vector sequence is used for the functional verification of the large designs (microprocessors). However, the generation of truly random test vectors for a BIST application is not very useful since the fault coverage would be different every time the test is performed as the generated test vector sequence would be different and unique every time.
- **Pseudo-Random Test Patterns:** These are the most frequently used test patterns in BIST applications. Pseudo-random test patterns have properties similar to random test patterns, but in this case the vector sequences are repeatable. The repeatability of a test vector sequence ensures that the same set of faults is being tested every time a test run is performed.

B. BIST Types

Logical BIST (LBIST): Today's SOCs can contain hundreds of memories, several types of logic, dozens of functional blocks obtained from diverse sources, and multiple clocks operating at multiple frequencies. This brings in its wake problems of defining a proper test strategy, optimizing test costs and the time required for testing. External access, embedded self-tests and diagnostics go a long way towards helping solve the test challenge. Logic BIST makes itself a valuable, if not an indispensable tool. One of LBIST's greatest virtues is its ability to test a circuit at its full operating speed. Something that even some very expensive external ATE systems cannot do. Further, "at-speed" testing, as it is called, has become essential for submicron chips.

LBIST is based on the scan method as the fundamental design for testability (DFT) technology. LBIST, which is designed for testing random logic, typically employs a PRPG to generate input patterns that are applied to the device's internal scan chain, and a MISR for obtaining the response of the device to these test input patterns.

Memory BIST (LBIST): With the advent of deep-submicron VLSI technology, core-based SOC design is attracting an increasing attention. On an SOC, popular reusable cores include memories, processors, input/output circuits, etc. Memory cores are obviously among the most universal ones - almost all system chips contain some type of embedded memory. However, to provide a low cost-cost test solution for the on-chip memory cores is not a trivial task. Digital systems are composed of data paths, control paths and memories. Defects in memory arrays are generally due to shorts and opens in memory cells, address decoder and read/write logic. These defects can be modeled as single and multi cell memory faults. The dominant use of embedded memory cores along with emerging new architectures and technologies make providing a low cost test solution for these on chip memories a very challenging task. The addition of extra circuitry to facilitate testing of memory chips, called DFT, or to allow the test mechanism to be completely contained within the chip, called BIST, has become of interest. MBIST, as its name implies, is used specifically for testing memories. It typically consists of test circuits that apply, read, and compare test patterns designed to expose defects in the memory device.

MBIST has been proven to be one of the most cost-effective and widely used solutions for memory testing for the following reasons: no external test equipment, reduced development efforts, tests can run at circuit speed to yield a more realistic test time, on-chip test pattern generation to provide higher controllability and observability. There now exists a variety of industry-standard MBIST algorithms, such as the walking 1/0 algorithm, march algorithm, the checkerboard algorithm, and the varied pattern background algorithm. With MBIST, the entire memory testing algorithm is implemented on-chip, and operates at the speed of the circuit, which are 2 to 3 orders of magnitude faster than a conventional memory test. Most MBIST schemes exploit the parallelism within the memory device to achieve a massive reduction in test time. This is done by a test mode where more than one memory cell is accessed with each address, usually by accessing the entire row of cells on a word line for a single read or write operation. For n cells in the memory, with $\sqrt{n}$ rows $\sqrt{n}$ columns, this reduces test time by a $\sqrt{n}$ factor. The memory BIST is similar to the basic BIST circuit, where CUT is replaced with the RAM. Additional block called controller is needed. A Controller is a hardware realization of a selected memory test algorithm, usually in the form of a Finite State Machine (FSM). The types of errors targeted in Logic BIST (LBIST) are

- Stuck – at – faults (SAF)
- Transition faults (targeted by at speed testing)

The types of errors targeted in memories (MBIST) are

- Stuck – At – Faults (SAF)
- Transition Faults (TF)
- Address Decoder Faults (ADF)
- Coupling Faults (CF)
- Data Retention Faults (DTF)

C. BISR

The VLSI manufacturing technology advances has made possible to put millions of transistors on a single die. This advancement in IC technology enabled the integration of all the components of a system into a single chip. A complex IC that integrates the major functional elements of a complete end product into a single chip is known as System on Chip (SOC). It enables the designers to move everything from board to chip. SOC incorporates a portable / reusable IP, Embedded CPU, Embedded Memory, Real World Interfaces, Software, Mixed-signal Blocks and Programmable Hardware. Reduction in size, lower power consumption, higher performance, higher reliability, reuse capability and lower cost are the benefits of using SOC. However, before SOC products can be widely seen on the market, many design and manufacturing issues have to be solved first. One of them is testing plural, heterogeneous cores of the SOC and the chip itself. With the trend of SOC technology, high density and high capacity embedded memories are required for successful implementation of the system. Memories are the widely used cores on the SOC products. Typically, many RAMs with various sizes are included in an SOC and they occupy a significant portion of the chip area. But, due to the continual advances in the manufacturing process of IC, provide designers the ability to create more complex and dense architectures and higher functionality on a chip.

Although it benefits the end user, but these manufacturing process advances are not without limitations. In particular, embedded high density memories in combination with these process limitations can result in poor over all yields. That is, RAMs have more serious problems of yield and reliability than any other embedded cores in an SOC. Depending upon the application and design, much of the low yield can be attributed to faults in the memory. Keeping the memory cores at a reasonable yield level is thus vital for SOC products. However, redundancy increases the silicon area and thus has a negative impact on yield. To maximize the yield, redundancy analysis is necessary. Conventionally, redundancy analysis (RA) is performed on the host computer of the ATE if it is an online process or on a separate computer if it is an offline process. Either way it is time consuming since RA algorithms are complicated and the memories that implement the redundancies are usually large. Moreover, embedded memories are harder to deal with Automatic Test Equipment (ATE). The BISR (Built in Self Repair) technique is a promising and popular solution for enhancing the yield of memories with the redundancy logic. Furthermore, redundancies of memories are not just for defects, redundancies can also be used to recover yield due to process variation in addition to yield recovery for defects.

Importance of Built-In Self-Test and Built-In Self-Repair: Traditionally memory repair is performed by using external equipment to test the memory, localize the faults and drive a laser beam that performs the repair. Electrical fuses or anti-

fuses can also be used to avoid the laser beam. From the view point of complexity, it is very difficult and costly to test the embedded memories from an external memory tester as the chip density continues to grow. Moreover, the accessibility of the embedded memories is low for external testers. Recent developments replace external equipment by BIST (Built in Self Test) and BISR schemes in order to maintain at reasonable levels the test and repair cost of embedded memories. An additional advantage is that the BIST and BISR can test and repair embedded memories at any time during the product life. This reduces the maintenance cost, and increases reliability and product life.

Redundancy Analysis: Memory repair relies on spare elements at different levels of design hierarchy. Faults can be compensated with redundant cells, rows, columns or even a complete memory block can be replaced. Many BISR schemes have been proposed for RAMs. BIRA (Built in Redundancy Analysis) algorithm is one key component of a BISR scheme, and it is responsible for allocating redundancies of RAMs under test. The BIRA modules can be used in parallel to allocate redundancy optimally. However, the numbers of BIRA modules are drastically increased with respect to the number of redundancies of RAMs under test. This results in very high area cost. The other strategy is, a single BIRA module is used to allocate redundancy optimally. The BIRA module is programmable and one repair strategy is programmed to allocate redundancies each time. Therefore, if the programmed repair strategy cannot repair the memory under test, then another repair strategy is programmed in the BIRA module and the memory under test is retested. This process is repeated until the successful repair strategy is found or all possible repair strategies are tried. Although the area cost of the BIRA circuit is reduced, this results in very high time cost of test and repair. Therefore, some of the existing BISR schemes use heuristic redundancy analysis algorithms to allocate redundancies of RAMs under test. In these algorithms, heuristics like “repair most” strategy is used, where resources are selected, such that the maximum numbers of faults are covered in each search step. These heuristic analysis algorithms attempt to optimize the repair efficiency and minimize the area cost and the time of the test/ repair.

III. BUILT IN REDUNDANCY ANALYSIS (BIRA)

As the density of memory has increased, the number of related defects has also increased. To increase device yield, many manufacturers use incorporated redundancy that can be used to replace faulty modules. Therefore, the implementation of effective redundancy algorithms is essential. Many redundancy analysis algorithms for performing redundancy allocation at ATE have been proposed. However, these algorithms are not adopted to be realized in built-in circuits. Thus, built-in redundancy-analysis (BIRA) algorithms which can cost-effectively be realized with built-in circuits are required for BISR schemes. The Figs.1 and 2 shows the memory array with different redundancy organizations.

A. Spare Row or Spare Column (1-D redundancy)

The memory contains spare rows or spare columns. When a fault is detected and must be repaired, the faulty row/column is replaced with one of the spare rows/columns. Although it is easier to repair faulty cells, however, this approach is inefficient since a whole spare row/column should be used for repairing a single faulty cell.

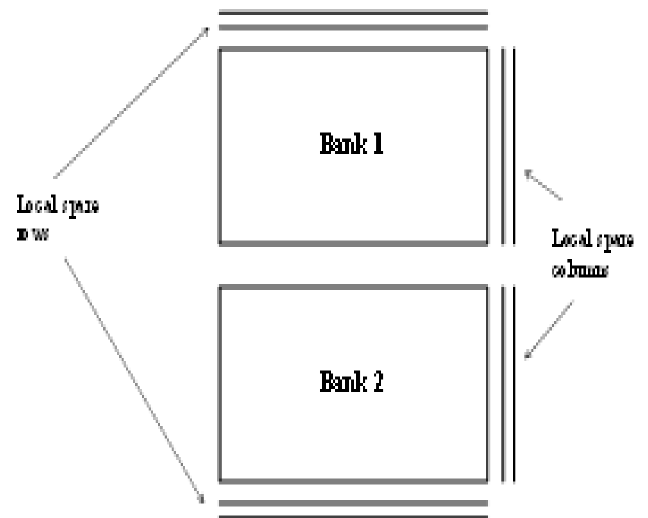

Fig.1.View of 1D redundancy.

B. Spare Row And Spare Column (2-D Redundancy)

With this approach, spare rows and spare columns are added in the memory array. Either a spare row or a spare column can be used to replace a faulty cell. This approach is more efficient than the first approach when multiple faulty cells are detected.

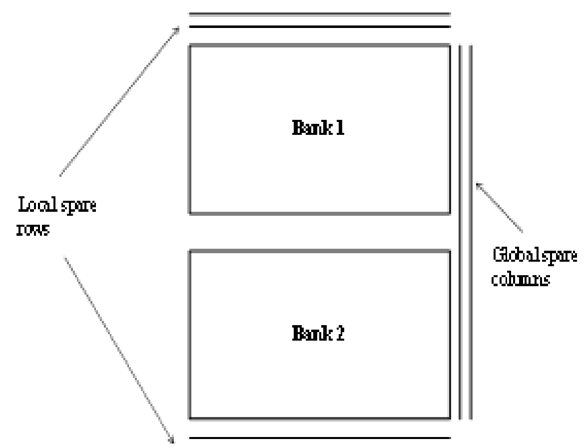

Fig.2. View of 2D redundancy Mmeory Bank.

However, the complexity for finding the optimal spare allocation is NP-complete. Moreover, owing to the high bandwidth of embedded RAMs, more spare columns and rows are required to achieve sufficient chip yield. This in turn increases the fabrication cost.

C. Selectable Redundancy

The proposed SRAM BISR strategy is flexible. The SRAM users can decide whether to use it by setting a signal. So the redundancy of the SRAM is designed to be selectable. In another word, some normal words in SRAM can be selected as redundancy if the SRAM needs to repair itself. We call these words Normal-Redundant words to distinguish them from the real normal ones.

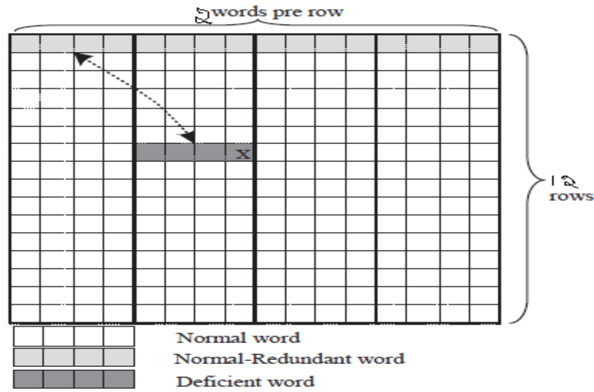

Fig.3.

This kind of selectable redundancy architecture can save area and increase efficiency. After BISR is applied, other modules in SRAM can remain unchanged. Thus the selectable redundancy won't bring any problem to SRAM module.

D. Programmable MBIST

Most of Memory BIST approaches concerns the programmability of the memory test algorithm. To enabling programmability of all components of memory test, test algorithm, test data, address sequence. The programmable memory BIST proposed has several advantages:

- It enables programming both test algorithms and test data.
- It implements test algorithm programmability at low cost, by extracting the different levels of hierarchy of the test algorithm and associating a hardware block to each of them, resulting on low cost hardware.
- It enables low-cost implementation of full-data programmability by adapting the transparent memory test approach in a manner that uses the memory under test for programming the test data.

The architecture for programming march test algorithms proposed in the fig.4. This architecture uses an instruction register specifying the current match test sequence by means of several fields indication.

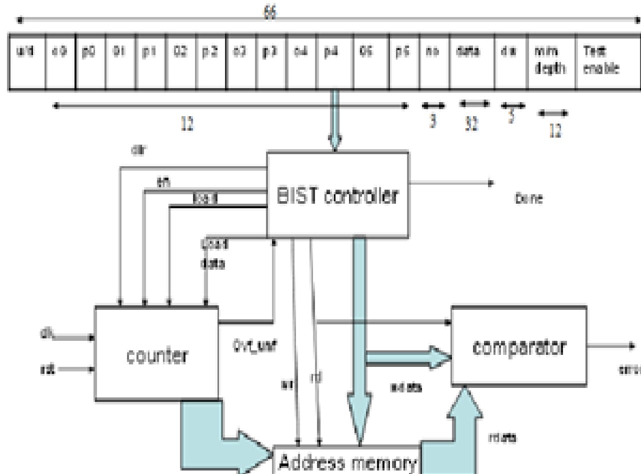

Fig.4.Block Diagram of programmable MBIST.

In the above block diagram, programmable MBIST contains 4 important parts.

- BIST controller
- Counter
- Comparator
- Address memory

BIST Controller: Built-in self-test (BIST) is a design technique that allows a circuit to test itself It is a set of structured-test techniques for combinational and sequential logic, memories, multipliers and other embedded logic blocks. The principle is to generate test vectors, apply them to the circuit under test or device under test, and then verify the response. Being an automated testing, BIST enables testing at high speed and high fault coverage. BIST controller coordinates the operations of different blocks of the BIST. Based on the test mode(TM) input to the controller, the system either operates in the normal mode or in the test mode. In this paper we explain an implementation of a restart able logic BIST controller for a combinational logic circuit using VHDL. It allows us to suspend the signature generation at any desired point in the test sequence. In this case, the BIST circuit is considered to comprise hold logic and a signature generation element. The hold logic will be implemented such that an external signal (HOED) can temporarily suspend signature generation in the signature generation element at specified times during the BIST session.

Counter: A counter that can change state in either direction, under the control of an up/down selector input, is known as an up/down counter. When the selector is in the up state, the counter increments its value. When the selector is in the down state, the counter decrements the count. A ring counter is a circular shift register which is initiated such that only one of its flip-flops is the state one while others are in their zero states. A ring counter is a Shift Register (a cascade connection of flip-flops) with the output of the last one connected to the input of the first, that is, in a ring. Typically, a pattern consisting of a single bit is circulated so the state repeats every n clock cycles if n flip-flops are used. It can be used as a cycle counter of n states. A Johnson counter (or switch tail ring counter, twisted-ring counter, walking-ring counter, or Amoebas counter) is a modified ring counter, where the output from the last stage is inverted and fed back as input to the first stage. The register cycles through a sequence of bit-patterns, whose length is equal to twice the length of the shift register, continuing indefinitely. These counters find specialist applications, including those similar to the decade counter, digital-to-analog conversion, etc. They can be implemented easily using D- or JK-type flip-flops.

Comparator: A method, apparatus and system of a self-test output for high density BIST are disclosed. In one embodiment, an integrated circuit includes one or more memories, a BIST controller coupled to the one or more memories to perform write operation and to receive a PASS/FAIL signal from each embedded memory and one or more comparators coupled to the one or more memories latch mutually identical outputted data coming from the memories upon a rising edge of an ORDY signal. In addition, the

comparators may compare the latched mutually identical outputted data and output associated PASS/FAIL signal to the BIST controller. The BIST controller registers the received PASS/FAIL result upon receiving the PASS/FAIL signal from the comparators. The integrated circuit may include output registers coupled to the BIST controller and the comparators output a data log substantially serially upon receiving a SHIFT/CLK signal from the BIST controller. A built-in self test (BIST) controller coupled to the one or more memories to perform write operation; and one or more comparators coupled to the one or more memories, wherein the one or more comparators latch identically addressable outputted data coming from the one or more memories upon a rising edge of an output ready (ORDY) signal, and wherein the one or more comparators compare the latched identically addressable outputted data and output associated PASS/FALL signal to the BIST controller, and wherein the BIST controller registers a PASS/FALL result upon receiving the PASS/FAIL signal from the one or more comparators.

Address Memory: In computing, memory address is a data concept used at various levels by software and hardware to access the computer's primary storage memory. Memory addresses are fixed-length sequences of bits conventionally displayed and manipulated as unsigned integers. Such numerical semantic bases itself upon features of CPU (such as the instruction pointer and incremental address registers), as well upon use of the memory like an array endorsed by various languages. Here are several types of memory addresses. In other words, a computer, and even one program may have several different memory address spaces.Digital computer's memory, more specifically main memory, consists of many MEMORY LOCATIONS, each having a physical address, a code, which the CPU (or other device) can use to access it. Generally only system software, i.e. the BIOS, operating systems, and some specialized utility programs (e.g., memory testers), address physical memory using machine code operands orprocessor registers, instructing the CPU to direct a hardware device, called the memory controller, to use the memory bus or system bus, or separate control, address and data busses, to execute the program's commands. The memory controllers' bus consists of a number of parallel lines, each represented by a binary digit (bit). The width of the bus, and thus the number of addressable storage units, and the number of bits in each unit, varies among computers.

IV. IMPLEMENTATION USING MARCH ALGORITHMS

The algorithms in most common use are the March tests. March tests have the advantage of short test time but good fault coverage. Test sequences or test algorithms for memories are known under the name of March tests. The test is "marching" through the memory. A March test consists of a sequence of March elements. A March element has a certain number of operations that must be applied to all memory cells of an array. The addressing order of a March element can be done in an up (↑), down (↓) way or (↕) if the order is not significant. A March primitive can be a write 1 (w1), write 0 (w0), read 1(r1) and read 0 (r0) that can be performed in a memory cell. The test is composed of March elements represented between (). March tests are the simplest tests (optimal) to detect most of the functional faults.

TABLE I: Comparison Of March Algorithms

Algorithms	Test length	Complexity	Fault coverage
March C-	10n	O(n)	AF, SAF, SOF,CF
March SS	22n	O(n)	AF, SAF, SOF, CF
M0M1	4n	O(n)	SAF, SOF
March LR with BDS	23n	O(n)	AF, SAF, SOF, CF

There are many March tests such as March C-, March SS, March0March1, March LR with BDS and so on. TABLE I compares the test length, complexity and fault coverage of them. ‘n’ stands for the capacity of SRAM. As shown in TABLE I, March C- has better fault coverage and shorter test time than any March algorithm. So March C- has been chosen as BIST algorithm in this paper. Its algorithm steps are as follows:

- March C- algorithm:

  {↕ (w0); ↑ (r0, w1); ↑ (r1, w0); ↓ (r0, w1);

  ↓ (r1, w0); ↕ (r0)}

Similarly, also observed the simulation results for,

- March M 0 M 1:

  {↕ (w0, r0); ↕ (w1, r1)}

- March SS(Simple Static):
  { ↕(w0); ↑(r0,r0,w0,r0,w1); ↑(r1,r1,w1,r1,w0);

  ↓(r0,r0,w0,r0,w1); ↓(r1,r1,w1,r1,w0); ↕(r0)}

- March LR(Realistic Linked) with BDS (Background Data Sequence):

  {↕ (w00); ↓ (r00, w11);

  ↑ (r11, w00, r00, r00, w11); ↑ (r11, w00);

  ↑(r00,w11,r11,r11,w00); ↑(r00,w01,w10,r10);

  ↑(r10,w01,r01); ↑(r01)}

In above steps, “up” represents executing SRAM addresses in ascending order while “down” in descending order. March Test Algorithms Table lists several relevant March algorithms reported in the literature. Table Gives the fault coverage and the operation count of these March algorithms, which are also called irredundant algorithms (by removing any operation from the test, The targeted fault coverage will be reduced). To generate custom March algorithms for improved defect coverage in new process technologies, an effective methodology was proposed. March algorithms are very easy to implement in either software or hardware.

V. SIMULATION AND SYNTHEIS RESULT SIMULATION RESULTS

Simulation results of this paper is shown in bellow Figs. 5 to 12.

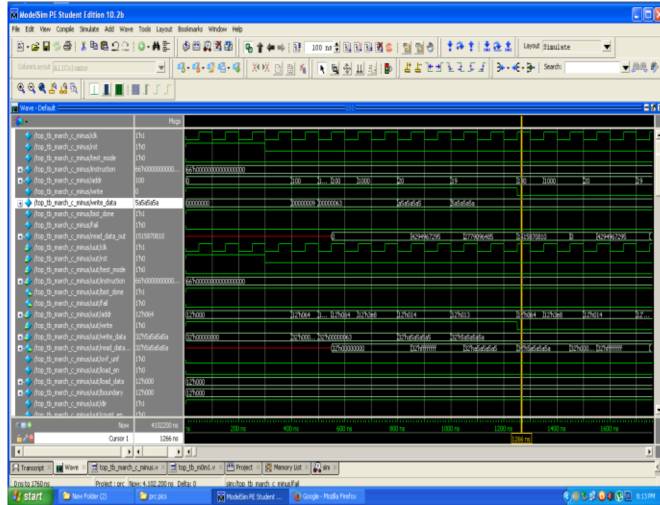

Fig.5.Simulation Result of March C alogorithm.

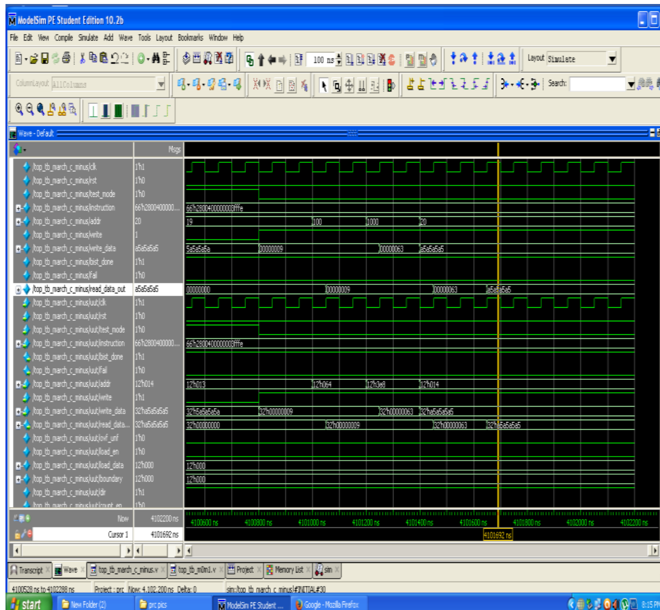

Fig.6. Simulation result of March C algorithm

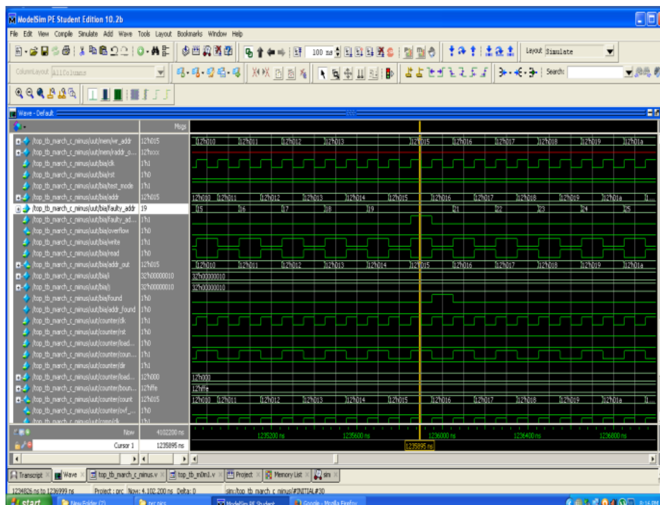

Fig.7. Simulation Result of March C algorithm

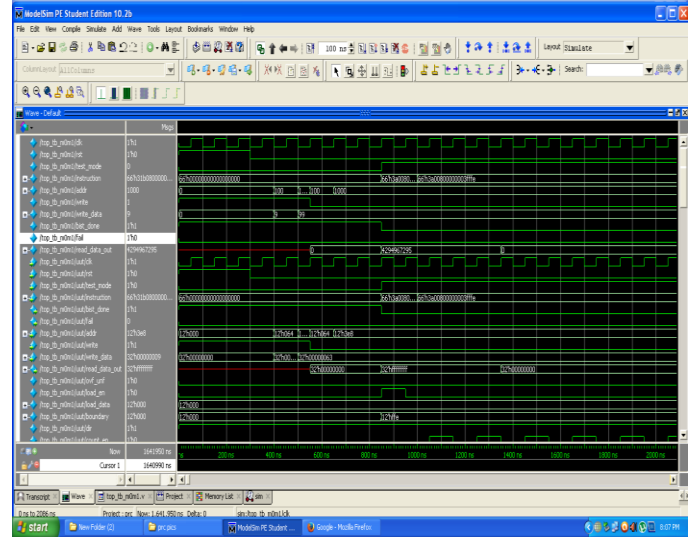

Fig.8. Simulation Result ofM0M1 Alogorithm

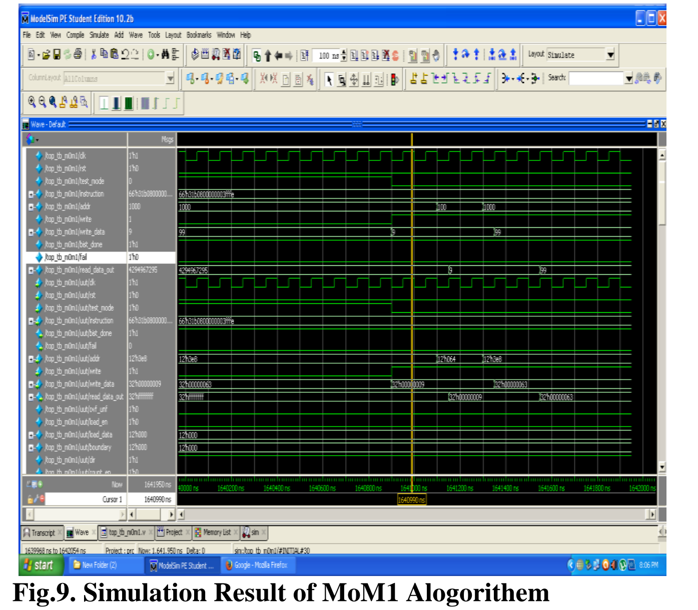

Fig.9. Simulation Result of MoM1 Alogorithem

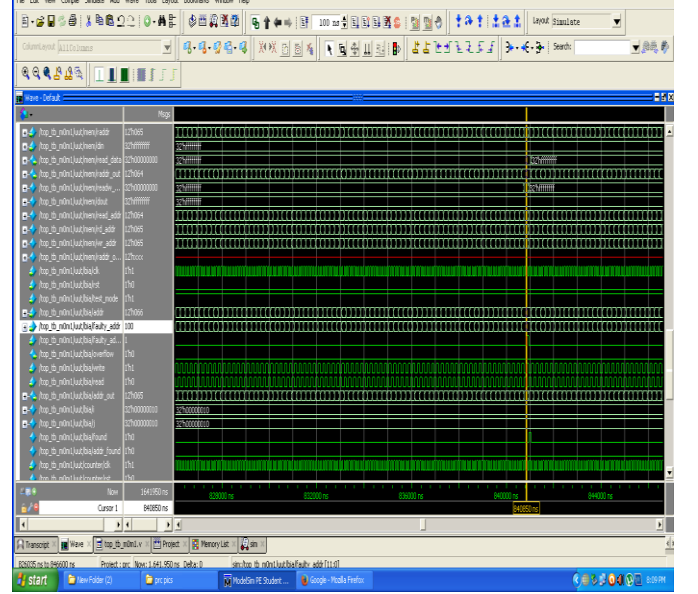

Fig.10. Simulation Result of M0M1 Algorithm

A. Synthesis Result

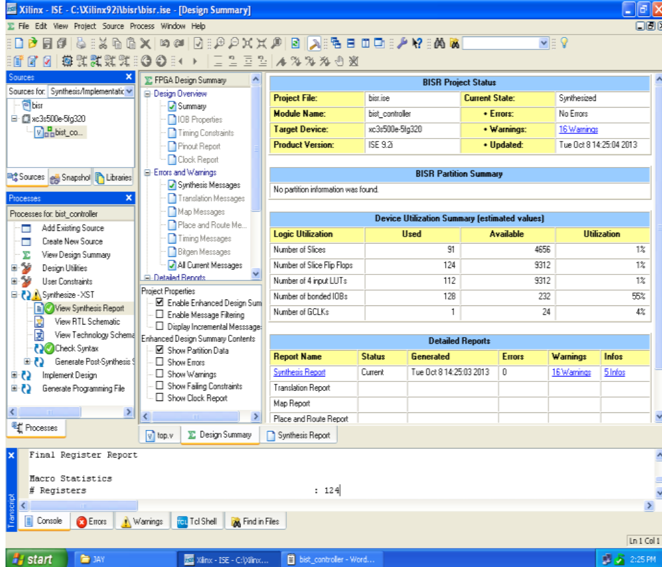

Fig.11.Sythesis Result of MBist

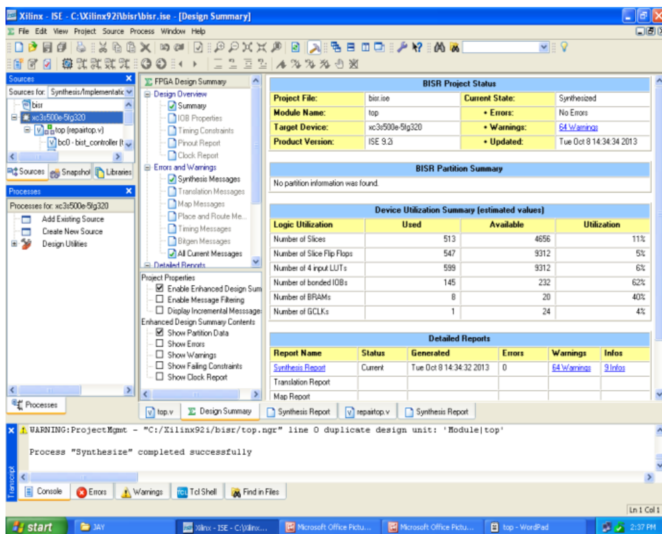

Fig.12.Sythesis Result of Proposed BISR

VI. CONCLUSION

An efficient BISR strategy for SRAM with programmable BIST and selectable redundancy has been presented in this paper. It is designed flexible that users can select operation modes of SRAM. The BIAA module can avoid storing fault addresses more than once and can repair fault address quickly. As these benefits are obtained at low area cost, the proposed memory BISR becomes highly attractive not only for test chips dedicated to new memory and/or process debug but also for integrating it into final products. The function of BISR has been verified by the post simulation. The BISR can work at up to 150MHz at the expense of 20% greater area. Future Scope: To increase the frequency of BISR system so that it can communicate with processor at equal frequency. To reduce the area required for the BISR system.

VII. REFERENCES

[1]Semiconductor Industry Association, "International technology roadmap for semiconductors (ITRS), 2003 edition," Hsinchu, Taiwan, Dec.2003.

[2] C. Stapper, A. Mclaren, and M. Dreckman, "Yield model for Productivity Optimization of VLSI Memory Chips with redundancy and Partially good Product," IBM Journal of Research and Development, Vol. 24, No. 3, pp. 398-409, May 1980.

[3]W. K. Huang, Y. H. shen, and F. lombrardi, "New approaches for repairs of memories with redundancy by row/column deletion for yield enhancement," IEEE Transactions on Computer-Aided Design, vol. 9, No. 3, pp. 323-328, Mar. 1990.

[4] P. Mazumder and Y. S. Jih, "A new built-in self-repair approach to VLSI memory yield enhancement by using neuraltype circuits," IEEE transactions on Computer Aided Design, vol. 12, No. 1, Jan, 1993.

[5]H. C. Kim, D. S. Yi, J. Y. Park, and C. H. Cho, "A BISR (built-in self- repair) circuit for embedded memory with multiple redundancies," VLSI and CAD 6th International Conference, pp. 602-605, Oct. 1999.

[6] Shyue-Kung Lu, Chun-Lin Yang, and Han-Wen Lin, "Efficient BISR Techniques for Word-Oriented Embedded Memories with Hierarchical Redundancy," IEEE ICIS-COMSAR, pp. 355-360, 2006.

[7]C. Stroud, A Designer's Guide to Built-In Self-Test,Kluwer Academic Publishers, 2002.

[8]Karunaratne. M and Oomann. B, "Yield gain with memory BISR-a case study," IEEE MWSCAS, pp. 699-702, 2009.

[9]I. Kang, W. Jeong, and S. Kang, " High-efficiency memory BISR with two serial RA stages using spare memories," IET Electron. Lett., vol. 44, no. 8, pp. 515-517, Apr. 2008.

[10]Heon-cheol Kim, Dong-soon Yi, Jin-young Park, and Chang-hyun Cho, "A BISR (Built-In Self-Repair) circuit for embedded memory with multiple redundancies," in Proc. Int. Conf. VLSI CAD, Oct. 1999.

[11] M. Sachdev, V. Zieren, and P. Janssen, " Defect detection with transient current testing and its potential for deep submicron CMOS ICs," IEEE International Test Conference, pp. 204-213, Oct. 1998.

[12]Mentor Graohics, MBISTArchitect Process Guide, Software Version 8.2009_3, Aug 2009, pp. 113-116.

[13]"Efficient Built in self repair strategy for embedded sram using selectable redundancy," Huamin Cao, Ming Liu, Hong Chen, Xiang Zheng, Cong Wang and Zhihua Wang.

[14]"Memory BIST with Address Programmability," Aymen Fradi Michael Nicolaidis, Lorena Anghel.