International Journal of Electrical and Electronics Research ISSN 2348-6988 (online)
Vol. 2, Issue 3, pp: (266-273), Month: July - September 2014, Available at: www.researchpublish.com

Programmable Built In Self-Test and Repair Analyzer Using Redundancy for Word Oriented SRAM

[1]Soni Mishra, [2]Sandeep Sunkari

[1,2]Department of Electronics and Communication Engineering, Prasad Engineering College, Jangaon, Telangana, India

Abstract: **Programmable Built-In Self-Test (PBIST) is a programmable testing technique where testing is accomplished without the aid of external hardware. The Repair Analyzer enables the fault bits to do the error correction using Redundancy strategy. To increase the reliability and yield of embedded memories, many redundancy mechanisms have been proposed. All the redundancy mechanisms bring penalty of area and complexity to embedded memories design. Considered that SRAM is used to configure for different needs, the BISR had better bring no change to other modules in SRAM. To solve the problem, a new redundancy scheme is proposed in this project. Some normal words in embedded memories can be selected as redundancy instead of adding spare words, spare rows, spare columns or spare blocks. Built-In Self-Repair (BISR) with Redundancy is an effective yield-enhancement strategy for embedded memories. This project proposes an efficient BISR strategy which consists of a Programmable Built-In Self-Test (PBIST) module, a Built-In Address-Analysis (BIAA) module, Multiplexer (MUX) module and a SRAM module. 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. In BIAA module, fault addresses and redundant ones form a one-to-one mapping to achieve a high repair speed. Besides, instead of adding spare words, rows, columns or blocks in the SRAMs, users can select normal words as redundancy, since it supports word-oriented memories.**

Keywords: **Programmable Built-In Self-Test (PBIST), Built-In Address-Analysis (BIAA), Multiplexer (MUX), SRAM, full repair analyzer.**

I. INTRODUCTION

As the technology changes the area in System-on-Chip (SoC) over 96%, is expected to occupy by embedded memories. Thus, optimizing the memories yield plays a crucial role in SoCs. However, memory fabrication yield is limited largely by random defects, random oxide pinholes, random leakage defects, gross processing and assembly faults, specific processing faults, misalignments, and other faults and defects.

The cost of verification and test for nowadays circuits represents an important part of the total IC final price. Hence, the domain of test represents a cornerstone for the industry. Recently, the advances in semiconductor memory technologies have become more complex resulting in a rapidly increasing transistor per memory design.

New design techniques enable a higher memory capacity implementation on a fixed die size. However, larger memory capacities require more extensive testing. Inevitably memory testing time increases the fault model to effectively model the variety of physical failures that could occur because of interference between closely packed cells. On the other hand, the more compact size will produces more defects during chip manufacturing, pushing yields down.

To increase the reliability and yield of embedded memories, many redundancy mechanisms have been proposed. In both redundant rows and columns are incorporated into the memory array. In spare words, rows, and columns are added into the word-oriented memory cores as redundancy. All these redundancy mechanisms bring penalty of area and complexity to embedded memories design. To solve the problem, a new redundancy scheme is proposed in this project. Some normal words in embedded memories can be selected as redundancy instead of adding spare words, spare

rows, spare columns or spare blocks. avoid storing fault address more than once. This paper proposes an efficient BISR strategy which can store each fault address only once.

The rest of this paper is organized as follows. Section II outlines SRAM fault models, test algorithms. Section III introduces the proposed BISR strategy. We present the details of the proposed BISR strategy including the architectures, procedures and the features. In section IV, the experimental results. Finally, Section V concludes this paper.

1. Fault Models , Test Algorithms

A fault model is a systematic and precise representation of physical faults in a form suitable for simulation and test generation [11]. Applying the reduced functional model, SRAM faults can be classified as follows in [12]:

- SAF ---- Stuck At Faults
- SOF ---- Stuck Open Faults
- AF ---- Address Fault
- CF ---- Coupling Faults
  - Intraword Coupling Faults
  - Interword Coupling Faults
- DRF ---- Data Retention Faults
- TF ---- Transition Faults

The details of the fault models can be referred in [12]. They are the foundations of the memory test. An Efficient and economical memory test should provide the best fault coverage in the shortest test time. BIST is used to test memories in the paper and its precision is guaranteed by test algorithms. The algorithms in most common use are the March tests. March tests have the advantage of short test time but good fault coverage. 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.

TABLE I. COMPARISION OF DIFFERENT MARCH TESTS

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

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:

$\{\updownarrow(w0); \uparrow(r0, w1); \uparrow(r1, w0); \downarrow(r0, w1);$

$\downarrow(r1, w0); \updownarrow(r0)\}$

Similarly, also observed the simulation results for,

- March M 0 M 1:

  $\{\updownarrow(w0, r0); \updownarrow(w1, r1)\}$

- March SS(Simple Static):

$\{\updownarrow(w0); \uparrow(r0,r0,w0,r0,w1); \uparrow(r1,r1,w1,r1,w0);$

⇓(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.

II. PROPOSED BISR STRATEGY

A. Programmable Memory Bist

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.

As stated in the introduction, the aim is to extent the above programmable BIST scheme to enable programming the address sequence, thus enabling programmability of all components of memory test: test algorithm; test data; address sequence. The architecture for programming march test algorithms proposed, is shown in figure 1. This architecture uses an Instruction Register specifying the current march test sequence by means of several fields indicating: the number of operations (NO) performed by the current march sequence; the order Up or Down (U/D) in which the address counter is used; the address mode(@mode); several operation fields (O1, O2, … Om-1), each determining one of the operations used by the march sequence (usually read and write, but other operations like read-modify-write, no operation, etc are also supported if necessary); the value of the memory word (DATA) to be used during the march sequence (fewer bits that the actual memory word can be used and expended by the BIST circuitry to create the full word); one data polarity bit for each operation field (P1, P2, … Pm-1), specifying if the corresponding operation uses the data word in its direct or complemented form; in order to implement a retention test; a bit (TE) set to 1 only at the last instruction of the test algorithm to indicate that the test finishes at the end of the current march sequence.

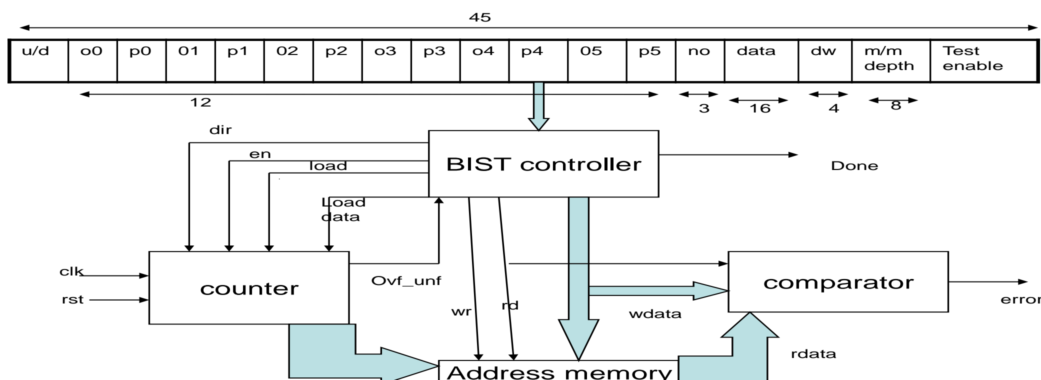


Figure 1: Programmable Memory Bist Architecture For March Test Algorithms.

International Journal of Electrical and Electronics Research ISSN 2348-6988 (online)
Vol. 2, Issue 3, pp: (266-273), Month: July - September 2014, Available at: www.researchpublish.com

B. Redundancy Architecture

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.

We take a 4K*32 SRAM, as shown in Figure 2. There are 4080 normal words and 16 Normal-Redundant words. When the BISR is used, the Normal-Redundant words are accessed as normal ones. Otherwise, the Normal-Redundant words can only be accessed when there are faults in normal words. In this case, the SRAM can only offer capacity of 4096 words to users.

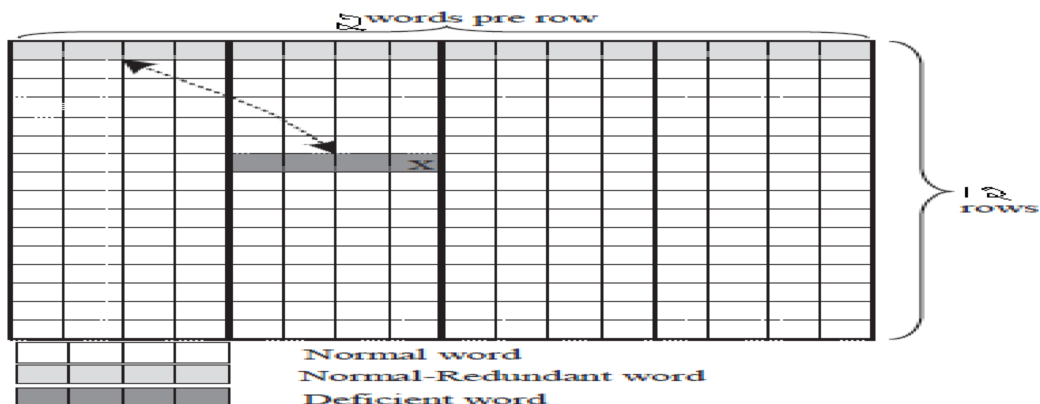


Figure 2: Architecture of Redundancy in Sram

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.

C. Overall BISR Architecture

The architecture of the proposed BISR strategy is shown in Figure3.

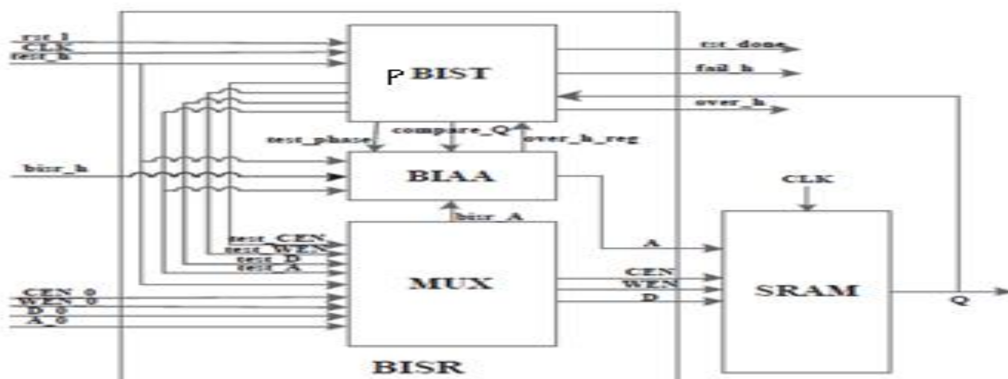


Figure 3: Proposed BISR Architecture.

It consists of three parts: PBIST module, BIAA module and MUX module. We call the SRAM with BISR a system. The BIST module uses March C- to test the addresses of the normal words in SRAM. It detects SRAM failures with a comparator that compares actual memory data with expected data. If there is a failure (compare_Q = 1), the current address is considered as a faulty address. The BIAA module can store faulty addresses in a memory named Fault_A_Mem. There is a counter in BIAA that counts the number of faulty addresses. When BISR is used (bisr_h = 1), the faulty addresses can be replaced with redundant addresses to repair the SRAM. The inputs of SRAM in different

operation modes are controlled by the MUX module. In test mode (bist_h=1), the inputs of SRAM are generated in BISR while they are equal to system inputs in access mode(bist_h= 0).

D. BISR Procedure

Figure 4 shows the proposed BISR block diagram.

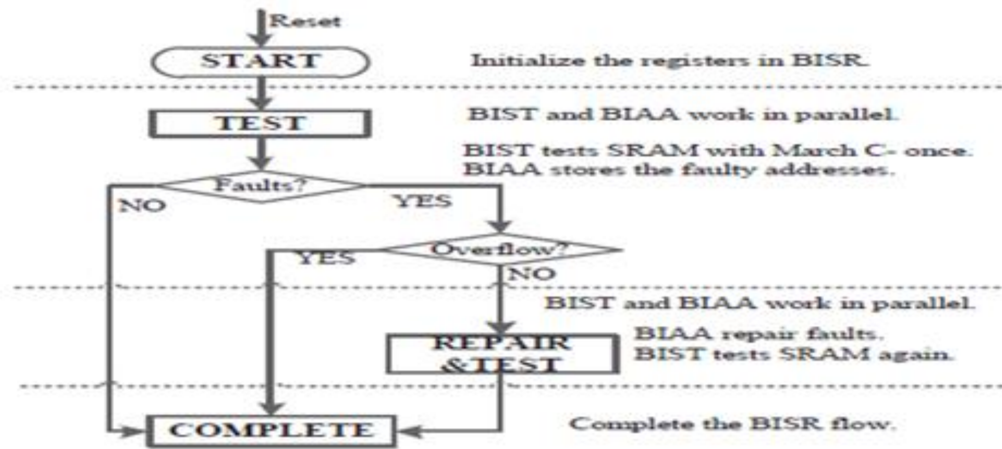


Figure 4. Block Diagram Of BISR

The BISR starts by resetting the system (rst_l = 0). After that if the system work in test mode, it goes into TEST phase. During this phase, the BIST module and BIAA module work in parallel. The BIST use March C- to test the normal addresses of SRAM. As long as any fault is detected by the BIST module, the faulty address will be sent to the BIAA module. Then the BIAA module checks whether the faulty address has been already stored in Fault-A-Mem. If the faulty address has not been stored, the BIAA stores it and the faulty address counter adds 1. Otherwise, the faulty address can be ignored. When the test is completed, there will be two conditions. If there is no fault or there are too many faults that overflow the redundancy capacity, BISR goes into COMPLETE phase. If there are faults in SRAM but without overflows, the system goes into REPAIR&TEST phase. The same as during TEST phase, the BIST module and BIAA module work at the same time in REPAIR&TEST phase. The BIAA module replaces the faulty addresses stored in Fault-A-Mem with redundant ones and the BIST module tests the SRAM again. There will be two results: repair fail or repair pass. By using the BISR, the users can pick out the SRAMs that can be repaired with redundancy or the ones with no fault.

E. Features Of The BISR

Firstly, the BISR strategy is flexible. TABLE II lists the operation modes of SRAM. In access mode, SRAM users can decide whether the BISR is used base on their needs. If the BISR is needed, the Normal-Redundant words will be taken as redundancy to repair fault. If not, they can be accessed as normal words.

TABLE II: SRAM OPERATION MODES

	Repair selection	Operation
Test mode (test_h=1)	Default: repair (bisr_h=1)	Access normal words. Repair faults and test.
	Don't repair (bisr_h=0)	Access normal words. Test only.
Access mode (test_h=0)	Repair (bisr_h=1)	Access normal words. Repair faults and write/read SRAM.
	Don't repair (bisr_h=0)	Access Normal- Redundant and normal Words. Write/read SRAM only.

Secondly, the BISR strategy is efficient. On one hand, the efficiency reflects on the selectable redundancy which is described as flexible above. No matter the BISR is applied or not, the Normal-Redundant words are used in the SRAM. It saves area and has high utilization. On the other hand, each fault address can be stored only once into Fault-A-Mem. As said before, March C- has 6 steps. In another word, the addresses will be read 5 times in one test. Some faulty addresses can be detected in more than one step. Take Stuck- at-0 fault for example, it can be detected in both 3rd and 5th steps. But the fault address shouldn't be stored twice. So we propose an efficient method to solve the problem in BIAA module. Figure shows the flows of storing fault addresses. BIST detects whether the current address is faulty. If it is, BIAA checks whether the Fault-A-Mem overflows. If not, the current fault address should be compared with those already stored in Fault-A-Mem. Only if the faulty address isn't equal to any address in Fault-A-Mem, it can be stored. To simplify the comparison, write a redundant address into Fault-A-Mem as background. In this case, the fault address can be compared with all the data stored in Fault-A-Mem no matter how many fault addresses have been stored.

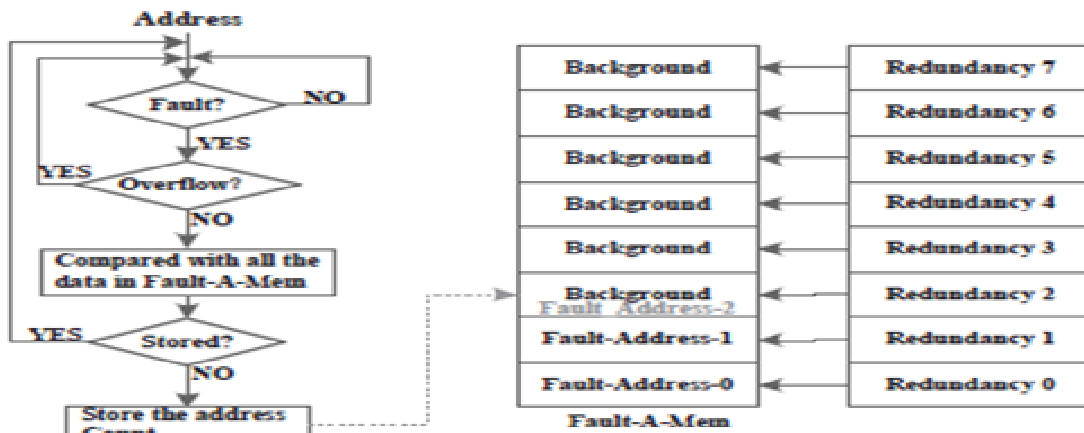


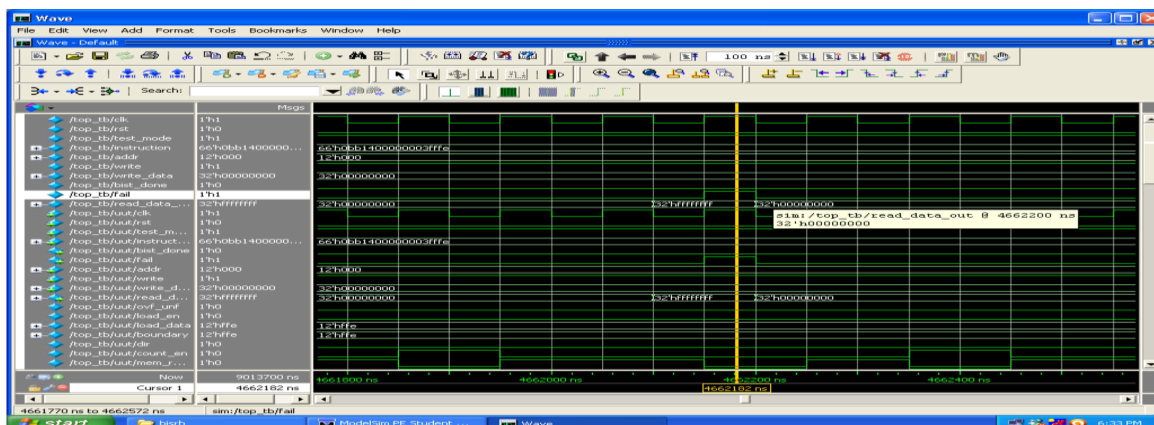
Figure 5. Flows Of Storing Fault Address

At last, the BISR strategy is high-speed. As shown in Figure 4, once a fault address is stored in Fault-A-Mem, it points to a certain redundant address. The fault addresses and redundant ones form a one-to-one mapping. Using this method, the BISR can quickly get the corresponding redundant address to replace the faulty one.

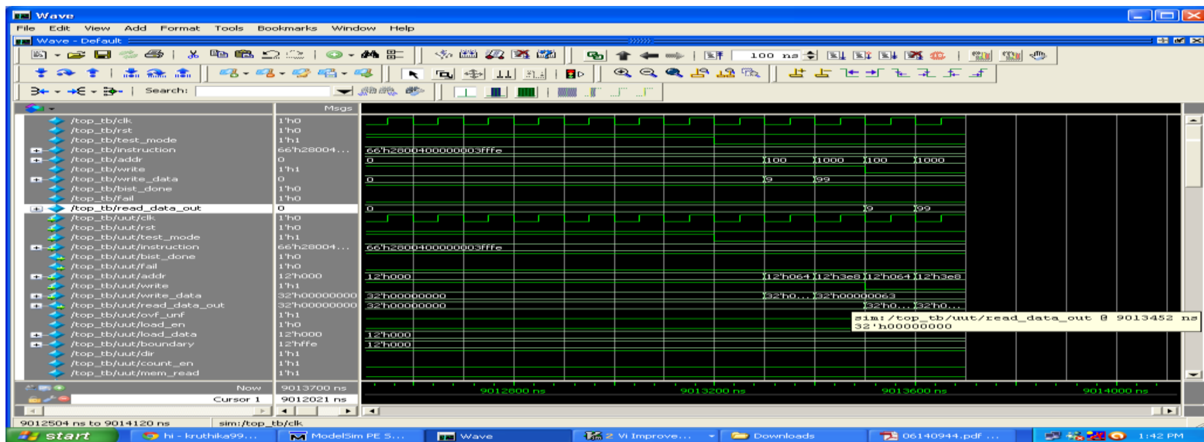
III. EXPERIMENTAL RESULTS

The proposed BISR was designed at RT level. The post simulation results show that the frequency of SRAM with BISR is at least 150MHz. To verify the function of BISR, a Stuck-at-0 fault, Stuck-at-1 fault, address fault and coupling fault (intra coupling fault) were set in the SRAM. The BISR scheme is synthesized using XILINX tool and simulation results using MODELSIM for March C-, March m0m1, March SS, March LR with BDS algorithms are observed. The simulation results for March C- algorithm applied for BISR strategy is observed as;

1. Simulation result for test enable = 1;

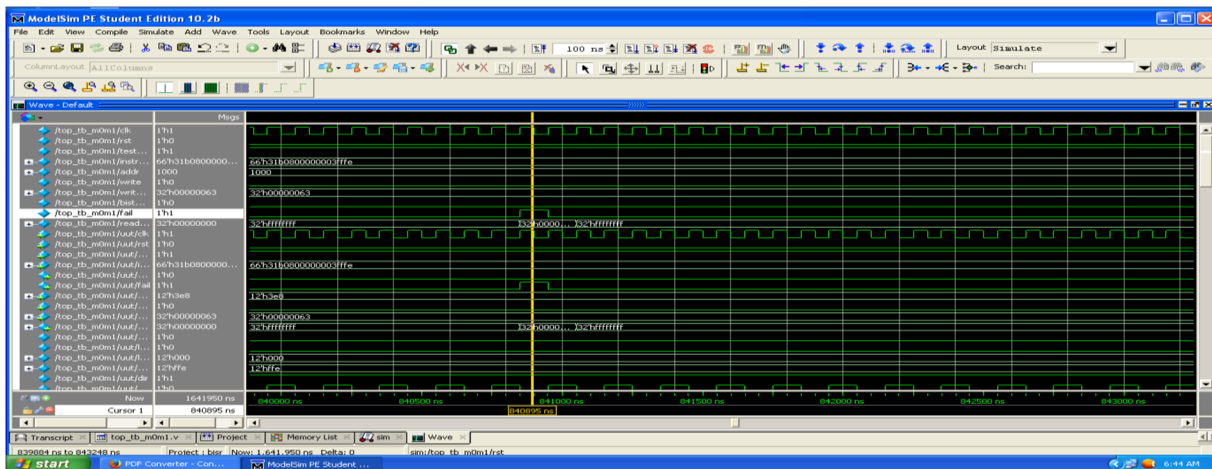


2. Simulation result for test enable = 0;

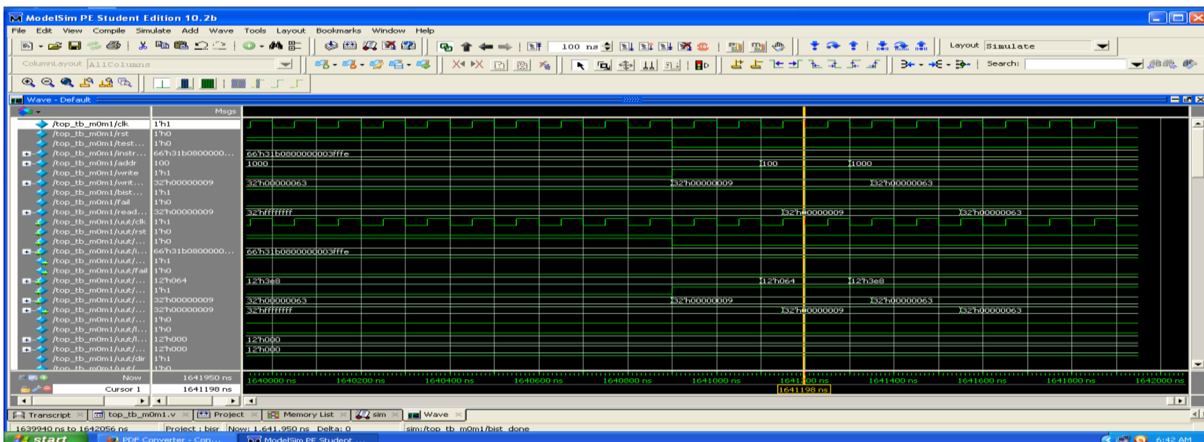


The simulation results for March0March1 algorithm applied for BISR strategy is observed as;

1. Simulation result for test enable = 1;



2. Simulation result for test enable = 0;



The low area cost required for implementing full data and full address programmability, integrating them in a product becomes attractive, as it can be very valuable in case of test quality crisis situations. Also, as test time is a lesser issue during debug of new memory design and/or fabrication process as well as during product ramp up, while test versatility becomes a must as fault complexity increases, the versatile BISR approach proposed in this paper becomes very attractive. BISR brings about 20% area penalties.

IV. CONCLUSIONS

Programmable Built in Self-Test and repair analyzer with different redundancy has been presented in this paper. It is designed flexible that users can select operation modes of word oriented 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 programmable 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 160MHz at the expense of 20% greater area.

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] 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.