International Journal of Computer Theory and Engineering, Vol. 2, No. 4, August, 2010
1793-8201

Modeling and Simulation of Multi-Operation Microcode-based Built-in Self Test for Memory Fault Detection and Repair

Dr. R.K. Sharma and Aditi Sood

***Abstract*—As embedded memory area on-chip is increasing and memory density is growing, problem of faults is growing exponentially. Considering the increasing impact that memory yield has on SoC yield in today's memory dominant SoCs, a high memory yield is required for acceptable levels of SoC yield. Thus memory fault modeling, detection and repair has come to take an important place. The architecture presented in this work implements the newly defined March SS algorithm which helps in detecting some recently modeled faults. Also, a word-oriented memory Built-in Self Repair methodology, which supports on-the-fly memory repair, is employed to repair the faulty locations indicated by the MBIST controller presented.**

***Index Terms*—Built-In Self Test (BIST), Built-In Self Repair (BISR), Defect-Per Million (DPM), Memory Built-in Self Test (MBIST), Microcoded MBIST, Memory Built-In Self Repair (MBISR).**

I. Introduction

According to the 2001 ITRS, today's system on chips (SoCs) are moving from logic dominant chips to memory dominant chips in order to deal with today's and future application requirements. The dominating logic (about 64% in 1999) is changing to dominating memory (approaching 90% by 2011) [1] as shown in Fig.1.

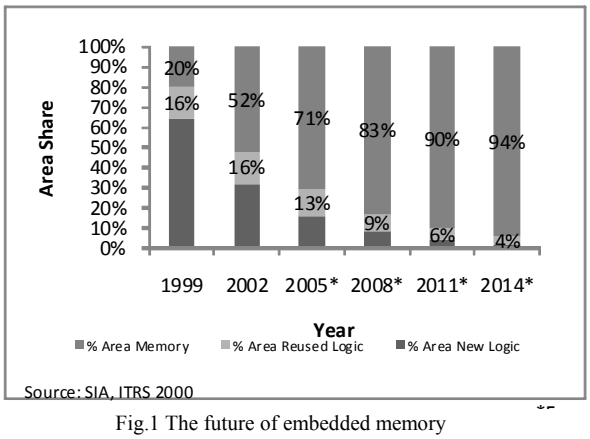

Fig.1 The future of embedded memory

The shrinking technologies give rise to new defects due to introduction of parasitic capacitances and resistive opens. These together with transistor short channel effect, cross talk effects, impact of process variation have to be necessarily taken into account for developing new fault models for embedded memories based on newer technologies. These fault models are then taken as basis of developing new, optimal, high coverage tests and diagnostic algorithms that allow for dealing with the new defects. The greater the fault detection and localization coverage, the higher the repair efficiency; hence higher the obtained yield. Memory repair is the necessary, since just detecting the faults is no longer sufficient for SoCs, hence both diagnosis and repair algorithms are required. Thus, the new trends in memory testing will be driven by the following items:

• Fault modeling: New fault models should be established in order to deal with the new defects introduced by current and future (deep-submicron) technologies.

• Test algorithm design: Optimal test/diagnosis algorithms to guarantee high defect coverage for the new memory technologies and reduce the DPM level.

• BIST: The only solution that allows at-speed testing for embedded memories.

BISR: Combining BIST with efficient and low cost repair schemes in order to improve the yield and system reliability as well.

A new microcoded BIST architecture is presented here which is capable of employing new test algorithms like March SS [5] and March RAW [3] that have been developed for coverage of some recently developed static and dynamic fault models. This BIST controller is then interfaced with word-oriented BISR circuitry as shown in the following Fig.2.

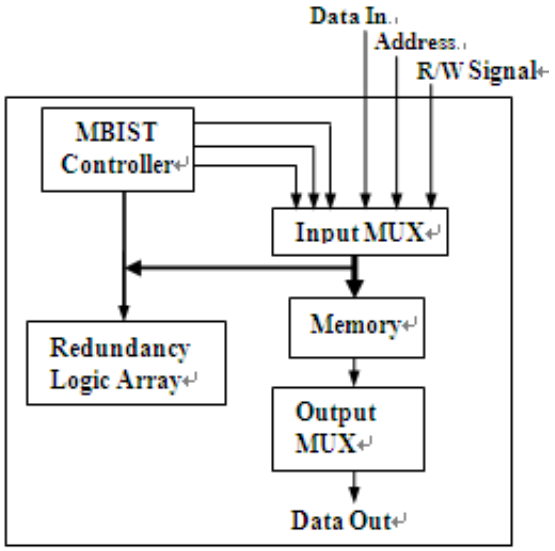

Fig.2 Principal Structure: MBIST and repair logic interface

Dr. R.K. Sharma and Aditi Sood are with the Department of Electronics and Communications Engineering, National Institute of Technology, Kurukshetra(email: mail2drrks@gmail.com, aditi.vlsi@gmail.com).

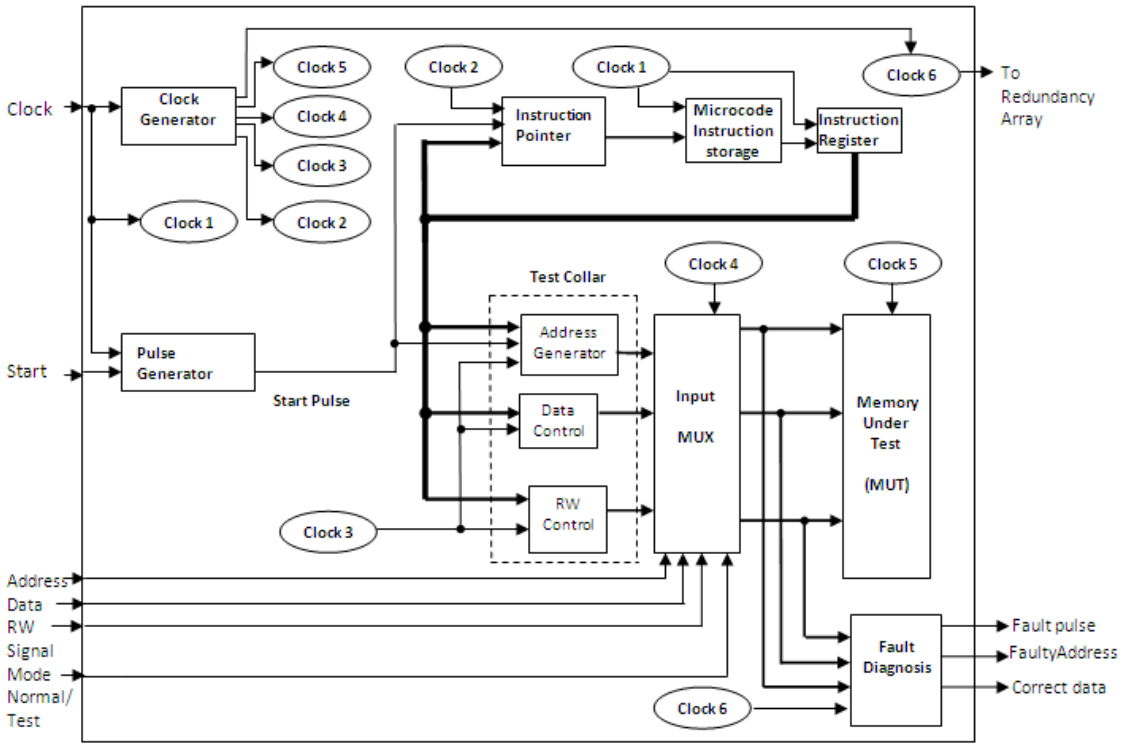

Fig.3. Microcode MBIST Controller Architecture

II. MICROCODE MBIST CONTROLLER

As shown in the previous section, the importance of developing new fault models increases with the new memory technologies.

Many well-known fault models, developed before late 1990s failed to explain the occurrence of complex faults. These included faults that were observed during the DPM screening of a large number of tests applied to a large number of memory chips. This implied that new memory technologies involving high density of shrinking devices lead to newer faults and stimulated the introduction of new fault models, based on defect injection and SPICE simulation. Write Disturb Fault (WDF), Transition Coupling Fault (Cft), Deceptive Read Disturb Coupling Fault (Cfdrd) etc. are examples of some such newly defined fault models [2]. Another class of faults called Dynamic faults which require more than one operation to be performed sequentially in time in order to be sensitized have also been defined. [3-4]

Traditional tests, like March C-, are thus becoming insufficient/inadequate for today's and the future high speed memories. Therefore, more appropriate test algorithms have been developed to deal with these new fault models like March SS and March RAW. March SS covers some of the new fault models like Deceptive Read Destructive fault (DRDF), Write disturb fault (WDF), etc., whereas March RAW covers some of the Dynamic faults.

Some of the recently developed architectures can perform up to two march operations per march element [6]. As a result, they are not capable to handle new test algorithms that involve as many as six/seven operations per march element. The proposed architecture has the ability to execute algorithms with unlimited number of operations per March element. Thus almost all of the recently developed March algorithms can be successfully implemented and applied using this architecture.

This has been illustrated in the present work by implementing March SS algorithm. The same hardware has been used to implement other new March algorithms. This requires just changing the Instruction storage unit, or the instruction codes and sequence inside the instruction storage unit. The instruction storage unit is used to store predetermined test pattern.

A) Methodology

The block diagram of the architecture is shown in Fig 3. The BIST Control Circuitry consists of Clock Generator, Pulse Generator, Instruction Pointer, Microcode Instruction storage unit, Instruction Register. The Test Collar circuitry consists of Address Generator, RW Control and Data Control.

Clock Generator produces simulated clock waveforms Clock2, Clock3, Clock4, Clock5, Clock6, for the rest of the circuitry based on the input clock (named Clock1) as shown in Fig. 4

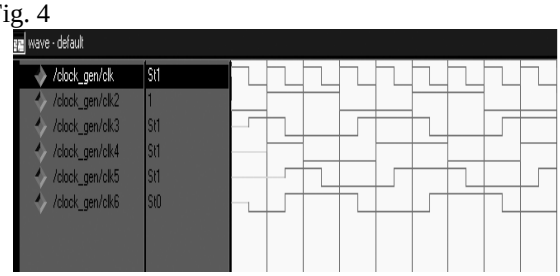

Fig 4. Simulated waveform of Clock generator Module

Pulse Generator generates a 'Start Pulse' at positive edge of the 'Start' signal marking the start of test cycle.

Instruction Pointer points to the next microword, that is the next march operation to be applied to the memory under test (MUT). Depending on the test algorithm, it is able to i) point at the same address, ii) point to the next address, or iii)

jump back to a previous address.

The flowchart in Fig. 5 precisely describes the functioning of the Instruction Pointer.

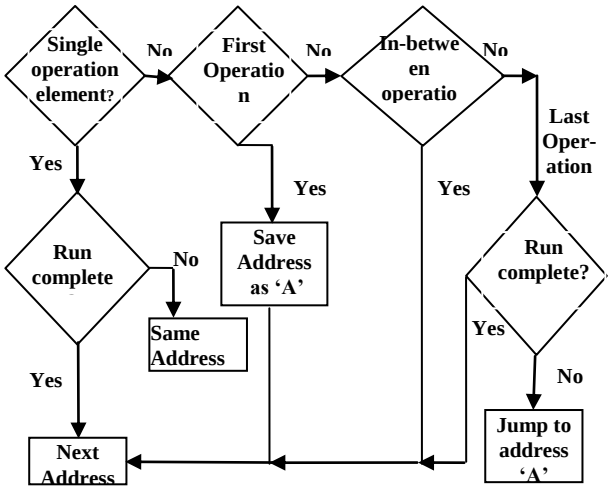

Fig. 5 Flowchart illustrating functional operation of Instruction Pointer

Here, 'Run complete' indicates that a particular march test operation has marched through the entire address space of MUT in increasing or decreasing order as dictated by the microcode.

Instruction Register holds the microword (containing the test operation to be applied) pointed at by the Instruction Pointer. The various relevant bits of microword are sent to other blocks from IR.

Address Generator points to the next memory address in MUT, according to the test pattern sequence. It can address the memory in forwards as well as backwards direction.

RW Control generates read or write control signal for MUT, depending on relevant microword bits.

Data Control generates data to be written to or expected to be read out from the memory location being pointed at by the Address Generator

The Address Generator, RW Control and Data Control together constitute the Memory *Test Collar*.

Input Multiplexer directs the input to memory by switching between test algorithm input and input given externally during the normal mode. The control signal for this multiplexer is also given externally by the user. If it indicates test mode then internally generated test data by BIST controller is given to the memory as input from the Test Collar. In case of Normal mode the memory responds to the external address, data and read/write signals.

Fault Diagnosis module works during the test mode to give the fault waveform which consists of positive pulses whenever the value being read out of the memory does not match the expected value as given by Test Collar. In addition, it also gives the diagnostic information like the faulty memory location address and the expected/correct data value.

This diagnostic information is used for programming the repair redundancy array.

B) Microcode Instruction specification.

The microcode is a binary code that consists of a fixed number of bits, each bit specifying a particular data or operation value. As there is no standard in developing a microcode MBIST instruction [7], the microcode instruction fields can be structured by the designer depending on the test pattern algorithm to be used.

The microcode instruction developed in this work is coded to denote one operation in a single microword. Thus a five operation March element is made up by five micro-code words. The format of 7-bit microcode MBIST instruction word is as shown in Fig. 6. Its various fields are explained as follows: Bit #1 (=1) indicates a valid microcode instruction, otherwise, it indicates the end of test for BIST Controller. Bits #2, #3 and #4 are used to specify first operation, in-between operation and last operation of a multi-operation March element, interpreted as shown in Fig. 6.

#1	#2	#3	#4	#5	#6	#7
Valid	Fo	Io	Lo	I/D	R/W	Data

Fo	Io	Lo	Description
0	**0**	**0**	**A single operation element**
1	**0**	**0**	**First operation of a Multi-operation element**
0	**1**	**0**	**In-between Operation of a Multi-operation element**
0	**0**	**1**	**Last Operation of a Multi-operation element**

Fig. 6 Format of Microcode Instruction word

Bit #5 (=1) notifies that the memory under test (MUT) is to be addressed in decreasing order; else it is accessed in increasing order. Bit #6 (=1) indicates that the test pattern data is to be written into the MUT; else, it is retrieved from the memory under test. Bit #7 (=1) signifies that a byte of 1s is to be generated (written to MUT or expected to be read out from the MUT); else byte containing all zeroes are generated.

The instruction word is so designed so that it can accommodate any existing or future March algorithm. The contents of Instruction storage unit for March SS algorithm are shown in Table 1.

The first march element M0 is a single operation element, which writes zero to all memory cells in any order, whereas the second march element M1 is a multi-operation element, which consists of five operations: i) R0, ii) R0, iii) W0, iv) R1 and v) W1. MUT is addressed in increasing order as each of these five operations is performed on each memory location before moving on to the next location.

TABLE 1

	#1 Valid	#2 Fo	#3 Io	#4 Lo	#5 I/D (0/1)	#6 R/W (0/1)	#7 Data (0/1)
M0: χ W0	1	0	0	0	0	1	0
M1: ↑{ R0	1	1	0	0	0	0	0
R0	1	0	1	0	0	0	0
W0	1	0	1	0	0	1	0
R1	1	0	1	0	0	0	0
W1}	1	0	0	1	0	1	1
M2: ↑ {R1	1	1	0	0	0	0	1
R1	1	0	1	0	0	0	1

W1	1	0	1	0	0	1	1
R1	1	0	1	0	0	0	1
W0	1	0	0	1	0	1	0
M3: ↓{R0	1	1	0	0	1	0	0
R0	1	0	1	0	1	0	0
W0	1	0	1	0	1	1	0
R0	1	0	1	0	1	0	0
W1}	1	0	0	1	1	1	1
M4: ↓{ R1	1	1	0	0	1	0	1
R1	1	0	1	0	1	0	1
W1	1	0	1	0	1	1	1
R1	1	0	1	0	1	0	1
W0}	1	0	0	1	1	1	0
M5: χ R0	1	0	0	0	1	0	0
	0	X	X	X	X	X	X

III. WORD REDUNDANCY MBISR

The BISR mechanism used here [17] employs an array of redundant words placed in parallel with the memory. These redundant words are used in place of faulty words in memory.

For successful interfacing with already existing BIST solutions as shown in Fig. 2, the following interface signals are taken from the MBIST logic:

1) A fault pulse indicating a faulty location address
2) Fault address
3) Expected data or correct data that is compared with the results of Memory under test

The MBISR logic used here can function in two modes

A) Mode 1: Test & Repair Mode

In this mode the input multiplexer connects test collar input for memory under test as generated by the BIST controller circuitry. As faulty memory locations are detected by the fault diagnosis module of BIST Controller, the redundancy array is programmed. A redundancy word is as shown in Fig. 7.

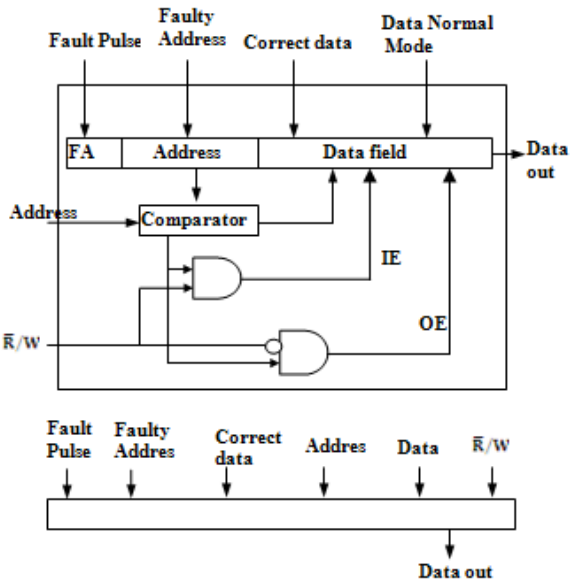

Fig.7. Redundancy Word Line

The fault pulse acts as an activation signal for programming the array. The redundancy word is divided into three fields. The FA (fault asserted) indicates that a fault has been detected. The address field of a word contains the faulty address, whereas the data field is programmed to contain the correct data which is compared with the memory output.

The IE and OE signals respectively act as control signals for writing into and reading from the data field of the redundant word.

An overflow signal indicates that memory can no longer be repaired if all the redundancy words have been programmed. The complete logic of programming of memory array is shown by Fig. 8.

B) Mode 2: Normal Mode

During the normal mode each incoming address is compared with the address field of programmed redundant words. If there is a match, the data field of the

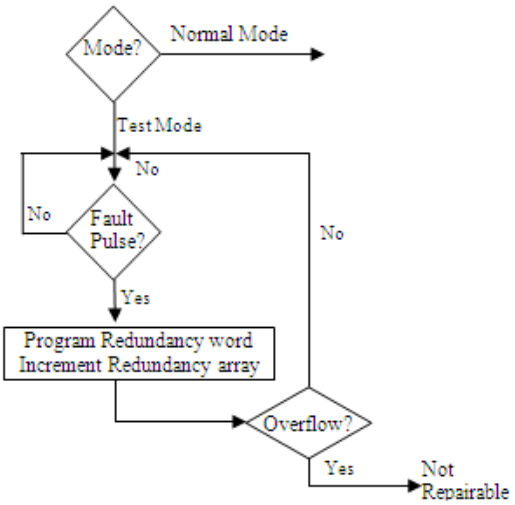

Fig. 7. Flowchart describing programming of redundancy array

redundant word is used along with the faulty memory location for reading and writing data. The output multiplexer of Redundant Array Logic then ensures that in case of a match, the redundant word data field is selected over the data read out ($\bar{R}/W = 0$) of the faulty location in case of a read signal. This can be easily understood by the redundancy word detail shown in Fig.7.

Fig.9 shows the Repair Module including the redundancy array and output multiplexer and its interfacing with the existing BIST module.

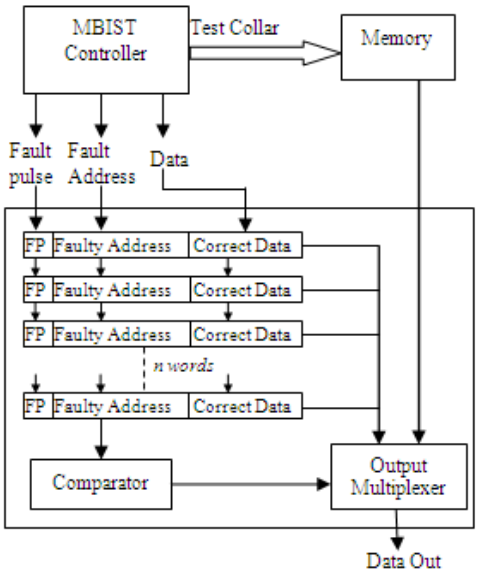

Fig. 8 Redundancy Array Logic

IV. Results

Mentor Graphic's ModelSim has been used to verify the functionality and timing constraints of Verilog coded BIST module, Repair redundancy array, and their interface.

The full architecture containing all these modules has been successfully synthesized using Xilinx ISE 8.2i.

The simulation waveform of a fault-free SRAM is shown in Fig.10.

The top module in the figure shows the interfacing of BIST Controller (including test collar), MUT and Comparator. As the START signal goes high, indicating the start of test, the first March element M0 of March SS algorithm is executed. As this is a write signal, no values are read out from the memory to be compared with expected or correct values and hence the output FAULT waveform of comparator is high impedance. As read operation starts at the beginning of execution of M1 element, the values from MUT are read out and compared with the expected values. The FAULT waveform shows a 'low' level throughout the test for a fault-free SRAM

The SRAM model is also amended to be in defective state by inserting faults. The simulated waveform is shown in Fig. 11.

The inserted faults are Deceptive Read Disturb fault (DRDF) at location 11, Write Disturb Fault (WDF) at location 13, Deceptive Read Disturb Coupling fault (CFdrd) at location 9 (victim) due to location 10 (aggressor), Write Disturb Coupling Fault (CFwd) at location 14 (victim) due to location 15 (aggressor) [9].

The fault detect waveform shows 12 pulses due to the above faults in given four locations, as the test elements march through MUT to uncover these defects.

The above stated faults cannot be detected by March C- algorithm but are easily detected by March SS Algorithm which has been implemented by the architecture presented in this work.

Fig. 12 is the simulated waveform of fault diagnosis module. Fault pulse, faulty location address and correct data signals are generated by this module for successful interfacing with the Redundancy Array logic. This is clearly illustrated by the simulated waveform (magnified) at the seventh fault pulse given by fault diagnosis module.

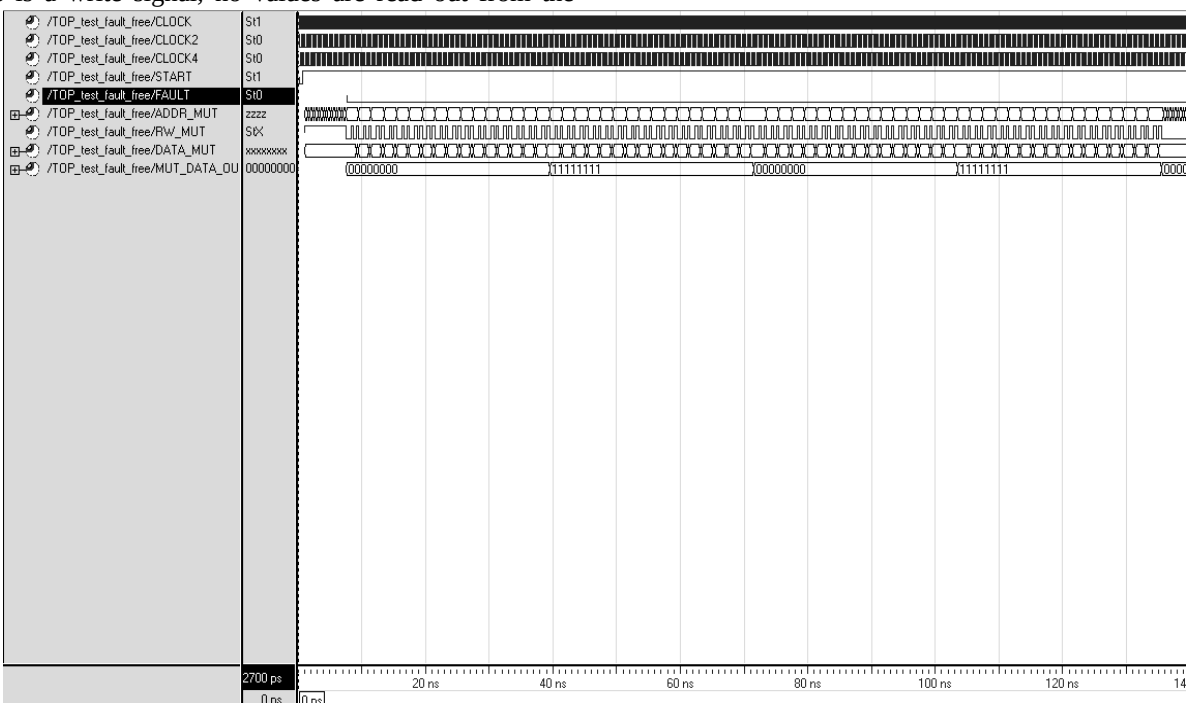

Fig. 10 Simulated waveform of fault-free SRAM

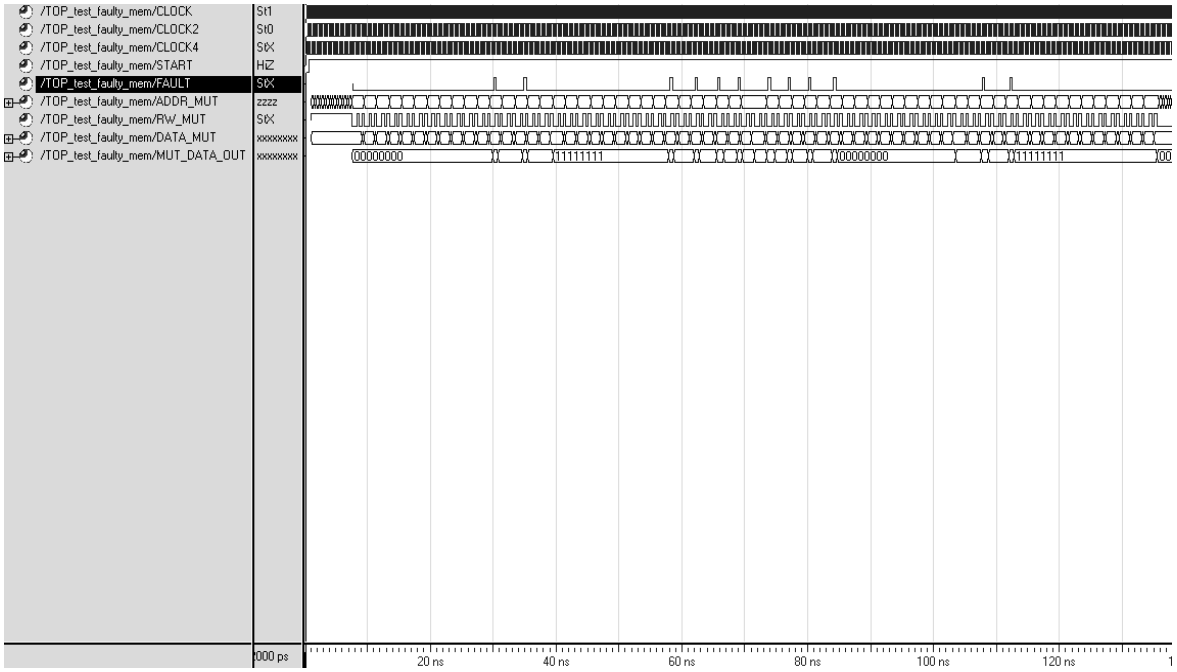

Fig. 11 Simulated waveform of faulty SRAM

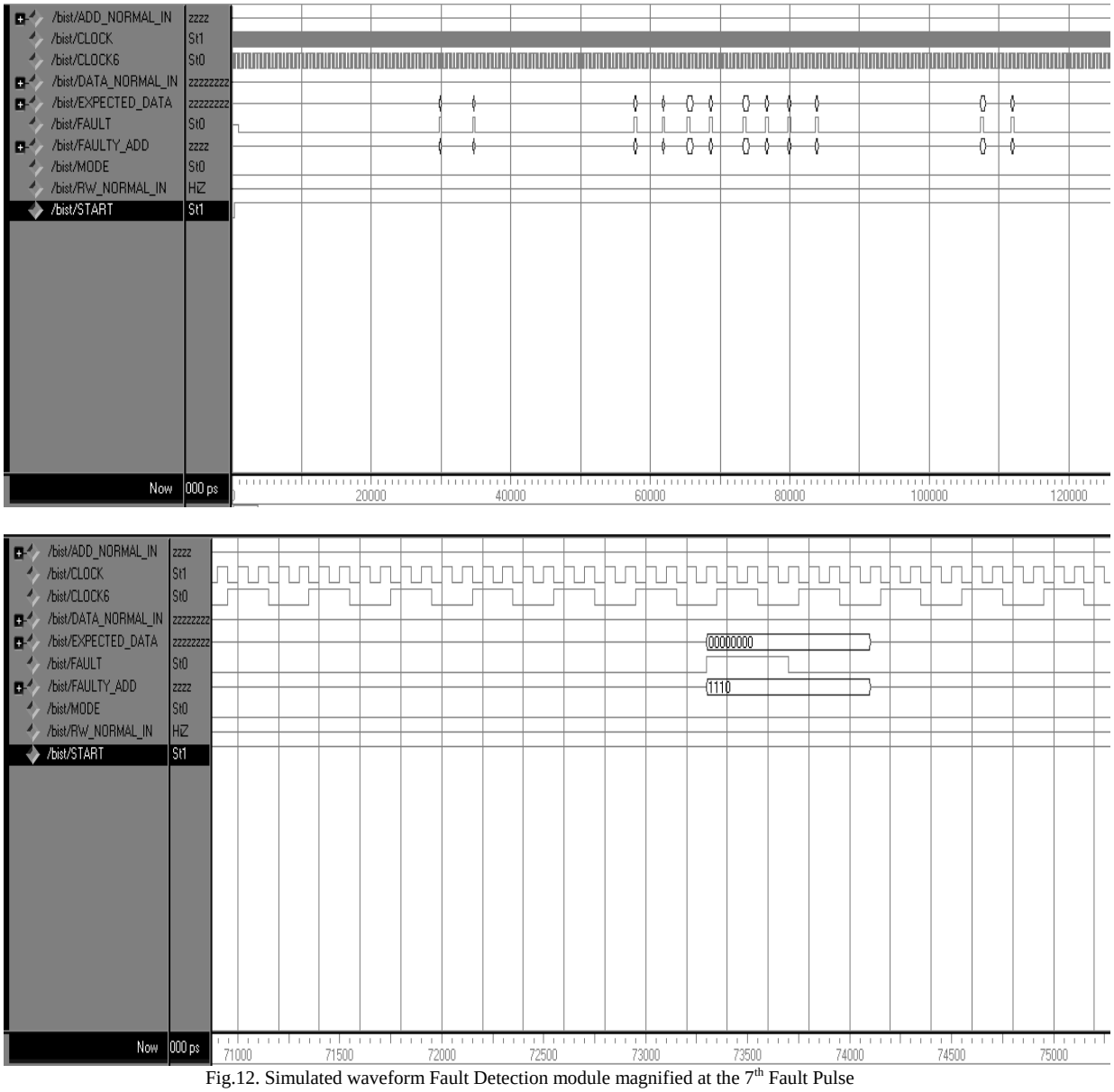

Fig.12. Simulated waveform Fault Detection module magnified at the 7th Fault Pulse

V. Conclusion

The simulation results have shown that the micro-coded MBIST architecture described here is an effective testing method to test embedded memories as it provides a flexible approach and better fault coverage. Just as March SS, any other new march algorithm can also be implemented using the same BIST hardware by changing the instructions in the microcode storage unit, without the need to redesign the entire circuitry. The word redundancy uses spare words in place of spare rows and columns. This repair mechanism avoids lengthy redundancy calculations as suggested by some other authors in their works [18], [19], as it stores faulty location addresses immediately supporting on-the-fly fault repair. Moreover, it can be interfaced easily with existing MBIST logic.

References

[1] International SEMATECH, "International Technology Roadmap for Semiconductors (ITRS): Edition 2001"

[2] S. Hamdioui, G.N. Gaydadjiev, A.J .van de Goor, "State-of-art and Future Trends in Testing Embedded Memories", *International Workshop on Memory Technology, Design and Testing (MTDT'04),* 2004.

[3] S. Hamdioui, Z. Al-Ars, A.J. van de Goor, "Testing Static and Dynamic Faults in Random Access Memories", *In Proc. of IEEE VLSI Test Symposium*, pp. 395-400, 2002.

[4] S. Hamdioui, et. al, "Importance of Dynamic Faults for New SRAM Technologies", *In IEEE Proc. Of European Test Workshop*, pp. 29-34, 2003.

[5] S. Hamdioui, A.J. van de Goor and M. Rodgers, "March SS: A Test for All Static Simple RAM Faults", *In Proc. of IEEE International Workshop on Memory Technology, Design, and Testing*, pp. 95-100, Bendor, France, 2002.

[6] N. Z. Haron, S.A.M. Junos, A.S.A. Aziz, "Modelling and Simulation of Microcode Built-In Self test Architecture for Embedded Memories", *In Proc. of IEEE International Symposium on Communications and Information Technologies* pp. 136-139, 2007.

[7] R. Dean Adams, "High Performance Memory Testing: Design Principles, Fault Modeling and Self-Test", Springer US, 2003.

[8] "Xilinx ISE 6 Software Manuals and help – PDF Collection",http://toolbox.xilinx.com/docsan/xilinx7/ books/manuals .pdf

[9] A.J. van de Goor and Z. Al-Ars, "Functional Fault Models: A Formal Notation and Taxonomy", *In Proc. of IEEE VLSI Test Symposium*, pp. 281-289, 2000.

[10] Zarrineh, K. and Upadhyaya, S.J., "On Programmable memory built-in self test architectures," *Design, Automation and Test in Europe Conference and Exhibition 1999*. Proceedings , 1999, pp. 708 -713

[11] Sungju Park et al, "Microcode-Based Memory BIST Implementing Modified March Algorithms", *Journal of the Korean Physical Society,* Vol. 40, No. 4, April 2002, pp. 749-753

[12] A.J. van de Goor, "Using March tests to test SRAMs", *Design & Test of Computers, IEEE,* Volume: 10, Issue: 1, March 1993 Pages: 8-14.

[13] R. Dekker, F. Beenker and L. Thijssen, "*Fault Modeling and Test Algorithm Development for StaticRandom Access Memories*", *Proc. IEEE Int. Test Conference,* Washington D.C., 1988, 343-352.

[14] R.Dekker, F. Beenker, L. Thijssen. "A realistic fault model and test algorithm for static random access memories". *IEEE Transactions on CAD,* Vol. 9(6), pp 567-572, June 1990.

[15] B. F. Cockburn: "Tutorial on Semiconductor Memory Testing" *Journal of Electronic Testing: Theory and Applications,* 5, pp 321-336 1994 Kluwer Academic Publishers, Boston.

[16] A.J. van de Goor, "Testing Semiconductor Memories, Theory and Practice" ComTex Publishing, Gouda, Netherlands.

[17] V. Schober, S. Paul, and O. Picot, "Memory built-in self-repair using redundant words," in Proc. Int. Test Conf. (ITC), Baltimore, Oct. 2001, pp. 995-1001.

[18] C.-T. Huang, C.-F. Wu, J.-F. Li, and C.-W. Wu, "Built-in redundancy analysis for memory yield improvement," IEEE Trans. on Reliability, vol. 52, no. 4, pp. 386-399, Dec. 2003.

[19] J.-F. Li, J.-C. Yeh, R.-F. Huang, and C.-W. Wu, "A built-in self-repair design for RAMs with 2-D redundancies," IEEE Trans. on VLSI Systems, vol. 13, no. 6, pp. 742-745, June 2005.