International Journal of Computer Applications (0975 – 8887)
Volume 78 – No.9, September 2013

Design and Implementation of Microcode based Built-in Self-Test for Fault Detection in Memory and its Repair

C. Padmini
Assistant Professor(Sr.Grade), ECE
Vardhaman college of Engineering, Hyderabad, INDIA

Ch.Tejdeep
M.Tech Student, ECE
Vardhaman college of Engineering, Hyderabad, INDIA

ABSTRACT

Memories are the most dominating blocks present on a chip. All types of chips contain embedded memories such as a ROM, SRAM, DRAM, and flash memory. Testing of these memories is a very tedious and challenging job as area over head, testing time and cost of the test play an important role. Embedded memories are occupying a significant portion of the System-on-chip area. Because of this trend and the nature of memory's small geometry, implementing a good memory testing strategy is one of the most significant decision making. Built-In Self-Test, a design technique which uses parts of the circuit to test the circuit itself is used for testing. BIST controller is used to control the total testing process of the memory.

General Terms

Built in self test (BIST); Built in self repair (BISR);

Keywords

Microcode BIST controller; Memory Built in self test (MBIST); Memory Built in self repair (MBISR)

1. INTRODUCTION

Embedded memory test design has become a substantial part of the System-on-chip development infrastructure. According to International technology roadmap for semiconductors (ITRS) recent report, memory cores will occupy around ninety percent of the area on chip. The yield of on-chip memories will dominate the chip yield. Built-in self-repair technique can increase the yield of memory from 5 % to 20 %, such that the soc yield can augment from 2% to 10% [1]. Embedded random access memory is one key component in modern complex system-on-chip design. Many RAM's with different sizes are included in a soc. These RAM's are subject to aggressive design rules causing them more prone to manufacturing defects. Testing has to done to ascertain the faults. Testing is no longer enough for embedded memories in the soc era. Repair mechanism BISR is required to repair these memories. BIST is usually used for only testing of these memories. Memory BIST is one of the most cost-effective and widely used solutions for memory testing for the following reasons. No external test equipment such as Automatic test equipment (ATE), reduced development efforts, test can run at circuit speed to yield a more realistic test time and on-chip test pattern generation for providing enhanced controllability.

March SS algorithm is used for detection of static and dynamic faults [4]. As it has good fault coverage it is used instead of other March algorithms such as March RAW, MATS, March Y, March B, March C- etc.

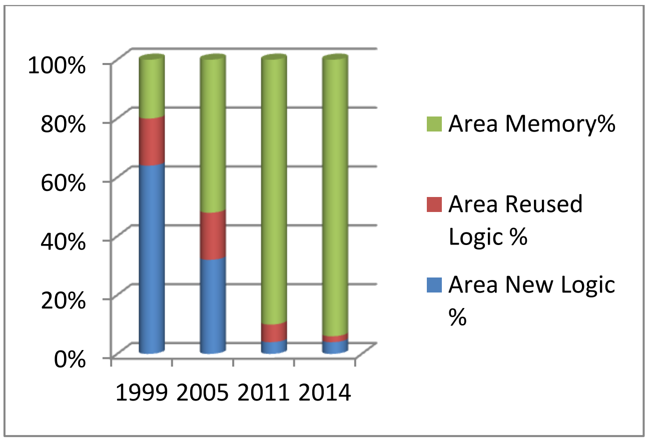

Figure1.Embedded memory area increasing on-chip every year

According to ITRS report the memory on-chip will increase as shown in the above figure [2]. Semiconductor Industry Association (SIA) published ITRS which has the detailed information about the updates to test equipment trends for nanometer designs.

2. BUILT-IN SELF-TEST

The basic BIST block diagram is shown below. It consists of a test pattern generator, unit under test and an output response analyzer.

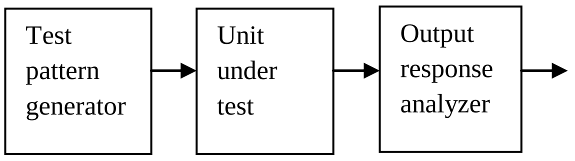

Figure2. BIST block diagram

A normal BIST block diagram is given in figure 2. It consists of a test pattern generator, unit under test and a response analyzer. The test pattern generates different test patterns which can be applied to the unit under test. The output response analyzer ascertains the faults in the unit under test and provides the faulty information as the output.

3. MARCH SS ALGORITHM

M0: ↕ (w0)

M1: ↑ (r0, r0, w0, r0, w1)

M2: ↑ (r1, r1, w1, r1, w0)

M3: ↓ (r0, r0, w0, r0, w1)

M4: ↓ (r1, r1, w1, r1, w0)

M5: ↕ (r0)

Notation of algorithm:

↑: address 0 to address n-1

↓: address n-1 to address 0

↕: either way

w0: write 0

w1: write 1

r0: read a cell whose value should be 0

r1: read a cell whose value should be 1

March SS algorithm is a finite sequence of march elements such as M0, M1, M2, M3, M4, M5. This is the dominant test algorithm implemented in the modern memory BIST. It has a test length of 22n [3]. Each element has multiple operations except the first and the last elements. They are single operation elements. March SS algorithm can detect faults like Write Disturb Fault (WDF), Deceptive Read Destructive Fault (DRDF), Data Retention Fault (DRF) [7] etc. Write disturb fault occurs in a cell when a non transition write operation causes a transition in the cell. Deceptive read destructive fault occurs in a cell if a read operation performed on the cell returns the correct logic value, while it results in changing the content of the cell.

4. MICROCODE BASED MBIST CONTROLLER

Micro code is nothing but a binary code. MBIST controller is designed to control the total test process that is performed on the memory. In figure 3 the MBIST controller along with the memory and the fault diagnosis block are shown. Clock generator, pulse generator, Instruction storage unit, Instruction register constitutes the BIST controller. This architecture has the ability to execute algorithms with any number of operations per march element[5]. The same hardware can be used to implement other type of march algorithms with a minimal changes in the instruction codes and the sequence inside the instruction storage unit. There are three types of BIST controllers. Microcode based BIST controller, Processor based BIST controller and Hard-wired BIST controller. When compared with the other two micro-code based BIST controller is compact and has low routing overhead and its test time is average. Hence micro-code based BIST controller is used in this paper.

4.1 Methodology

4.1.1 Clock Generator

It generates all clock signals from the base clock signal. A total of five clock signals are generated by the clock generator. Each clock signal has one fourth the base clock signal frequency. Clock4, Clock5 are delayed versions of Clock 3. Clock 6 is the inverse of Clock 3. Clock 3 is the inverse of Clock 2.

4.1.2Pulse Generator

It generates a ‘start pulse’, the pulse being high indicates that the test process has started and if it is low, indicates that the test process is not initiated.

4.1.3. Instruction Pointer

It points to the instructions that are present in the microcode instruction storage unit. Each march operation is expressed in terms of microcode. These operations are performed on the memory.

The below flowchar illustrate how the instruction pointer operates. When it is a single operation element then the pointer MBIST controller checks whether the operation is performed in all the memory locations. This is indicated by ‘Run Complete’. If the operation is performed on all the memory locations then the instruction pointer points to the next address which contains the next microcode instruction.

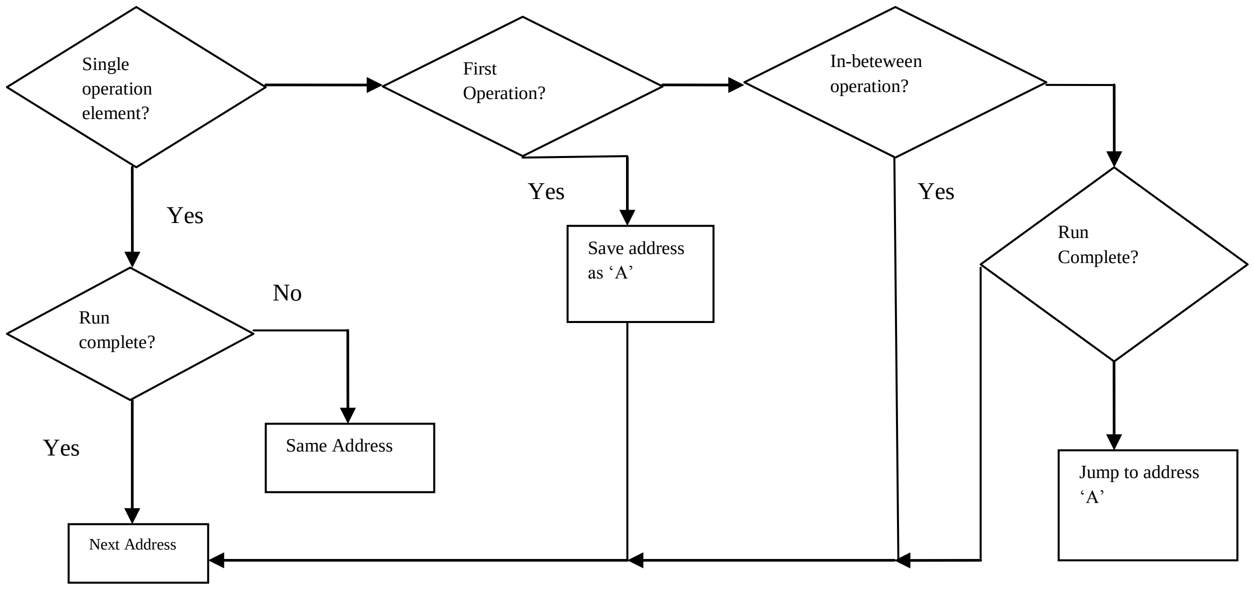

Figure 3. Flow chart representing the functional operation of Instruction pointer

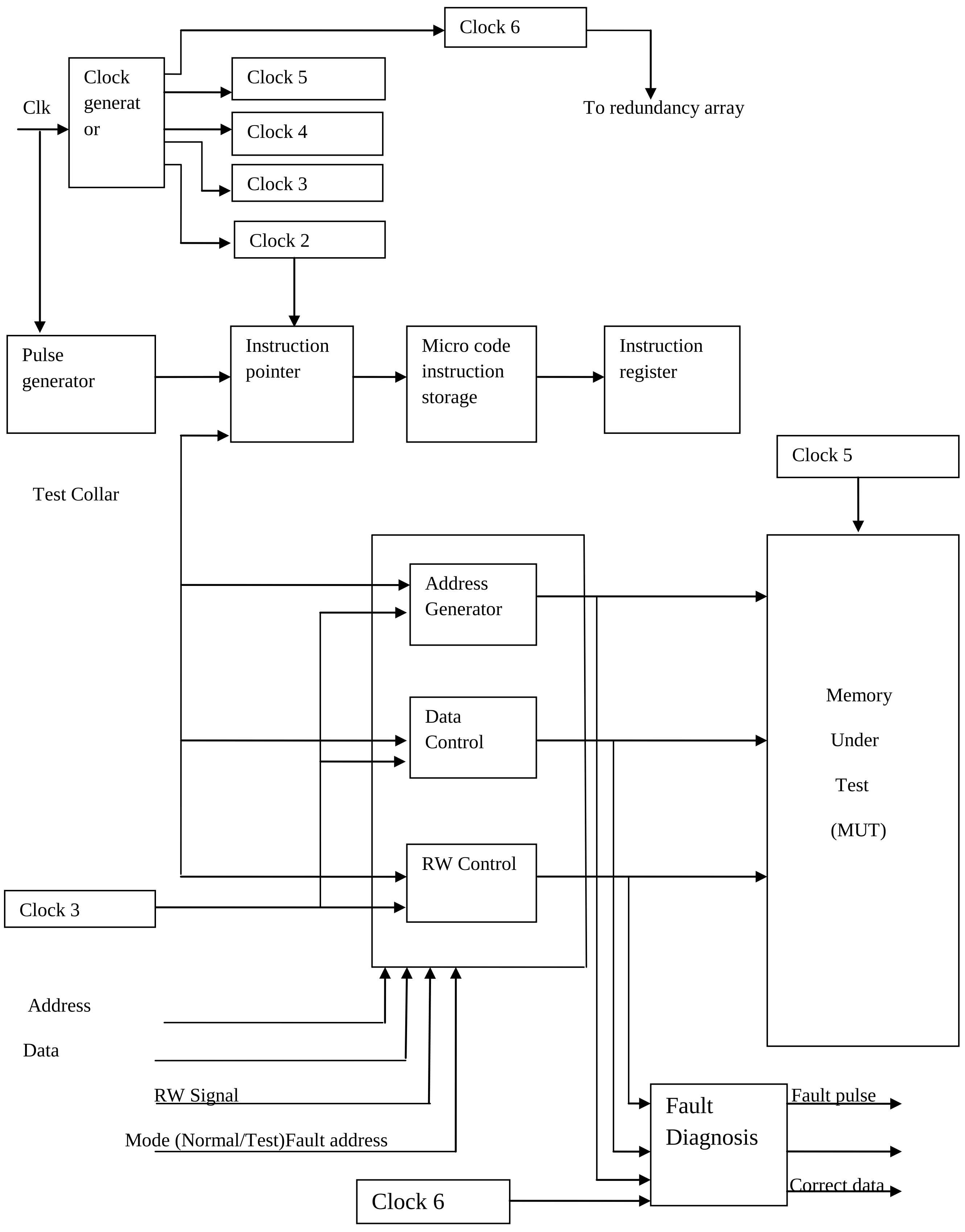

Figure4. Microcode MBIST controller and its interface with fault diagnosis

If the operation is not performed on all the memory locations then the instruction pointer points to the same address where the current executing instruction is present. If the element consists of multiple operations then the controller checks if it is the in-between operation or not and performs the operation on all the memory locations and then finally checks if there are any other in-between operations,if there aren't any, it points to the final operation and when it is done with the final operation,it points to the next microcode instruction or the next element of the March SS algorithm.

4.1.4 Instruction Register

It holds the microcode instructions i.e; the march algorithm operations.These instructions are pointed by the instruction pointer.

4.1.5 Test Collar

The address generator, data control and the RW control constitutes the test collar.

4.1.6 Address Generator

It points to the next address in the memory where the particular operation is performed. The operation can be read or write.

4.1.7 Data Control

This generates the data that is to be written/read in/from the address location generated by the address generator.

The MBIST controller can work in two modes. One is the Test mode and the other is the Normal mode. These modes can be indicated by using a multiplexer.

4.2 Format of Microcode instruction

A microcode is a binary code that consists of a fixed number of bits and each bit specifies a particular data or an operation value.There are no standards to develop a microcode hence it can developed according to the test algorithm[6]. So here the microcode is written according to the March SS algorithm.

Each microcode represents a single operation of the march ss algorithm. Each microcode consists of seven bits. The first bit represents whether the microcode instruction is valid or not, it is denoted by 'Valid' . The second bit repersents whether it is a first operation or not, denoted by 'Fo'. Each element of the algorithm consists of multiple operations except the first and the last elements of the algorithm. Hence these operations are identified by the second,third and the fourth bits of the microcodeinstruction. The third bit is for representing the in-between operations,denoted by 'Io'. The fourth bit for representing the last operation of the march element,denoted by 'Lo'. The fifth bit represents the order in which the data is written or read to/from the memory, denoted by 'I/D' . The sixth bit represents the read or write operation, denoted by 'R/W'. The seventh bit is represents the data that is to be written or read to/from the memory,it is denoted by 'Data'.

#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

Table 1.Format of Microcode instruction word

The above table shows the combination of Fo, Io and Lo and their representation of the operations.

The MBIST controller finds out the faults present in the memory when it is in the test mode. When it is in the normal mode, fault output will be obtained instead of the correct data that is written in to thememory as MBIST cannot repair the faulty memory locations. Hence MBISR is used to repair the faulty memory locations. The expected data or the correct data is compared with the data that is obtained as the result from the memory under test ,when the MBIST in the test mode, so it obtains the location address or the word address in which the fault is present and shows the fault memory location in the output. In the normal mode it takes input and gives the faulty output as result from the memory under test from the faulty address locations.

The below table shows the microcode for each and every operation of the march ss algorithm.

	Valid	Fo	Io	Lo	I/D	R/W	Data
M0: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	0	0
R0	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:R0	1	0	0	0	1	0	0
	0	X	X	X	X	X	X

5. WORD ORIENTED MEMORY BUILT-IN SELF-REPAIR (MBISR)

The MBISR consists of redundant array of words placed in parallel with the memory. These redundant words are used instead of the faulty word present in the memory.

MBISR operates in two modes just as MBIST operates in two modes. One is test and repair mode the other is normal mode. MBISR in the test mode finds out the faulty location addresses and stores them in the spare or redundant words. In the normal mode when the data is written in the memory, the data that is written in to the faulty locations is also written in the redundant words. This redundant word is divided in to three fields. The fault asserted (FA) field of the word indicates that a fault is detected, the address field of the word stores the address of the faulty location address and the data field contains the correct data that is written in to the memory.

FA	Address	Data Field

Figure 4.Redundant word format

When the MBISR is working in the normal mode, the data that is written in to the faulty memory location is not read as output, instead the data from the redundacy word or the spare word is taken as output. For this a multiplexer can be used and the output from memory or the redundant word is selected. A comparator is used for comparing the address from the memory location that is read and the address that is stored in the redundant word. If both the addresses are same then the data present in the redundant word is selected by multiplxer.

6. BEHAVIOUR SIMULATION

Verilog HDL is used to write the code for the above architecture and synthesized using Xilinx ISE 12.1 tool.

7. RESULTS

The simulation results of the MBIST controller in both test and normal mode are shown in the figures below.

In the figure 5, the fault is present in the fifth and the sixth locations hence the fault pulse is high.

In the figure 6, the outputs from the fifth and the sixth locations are not the same as the values that are given as input to those locations. Hence the faulty outputs are obtained.

In the figure 7, the Addressfield is showing the locations in which the faults are present i.e., the fifth and the sixth locations.

In the figure 8, the memout is giving the correct output from the memory. The Datafield is storing the data that is actually written in to the faulty locations.

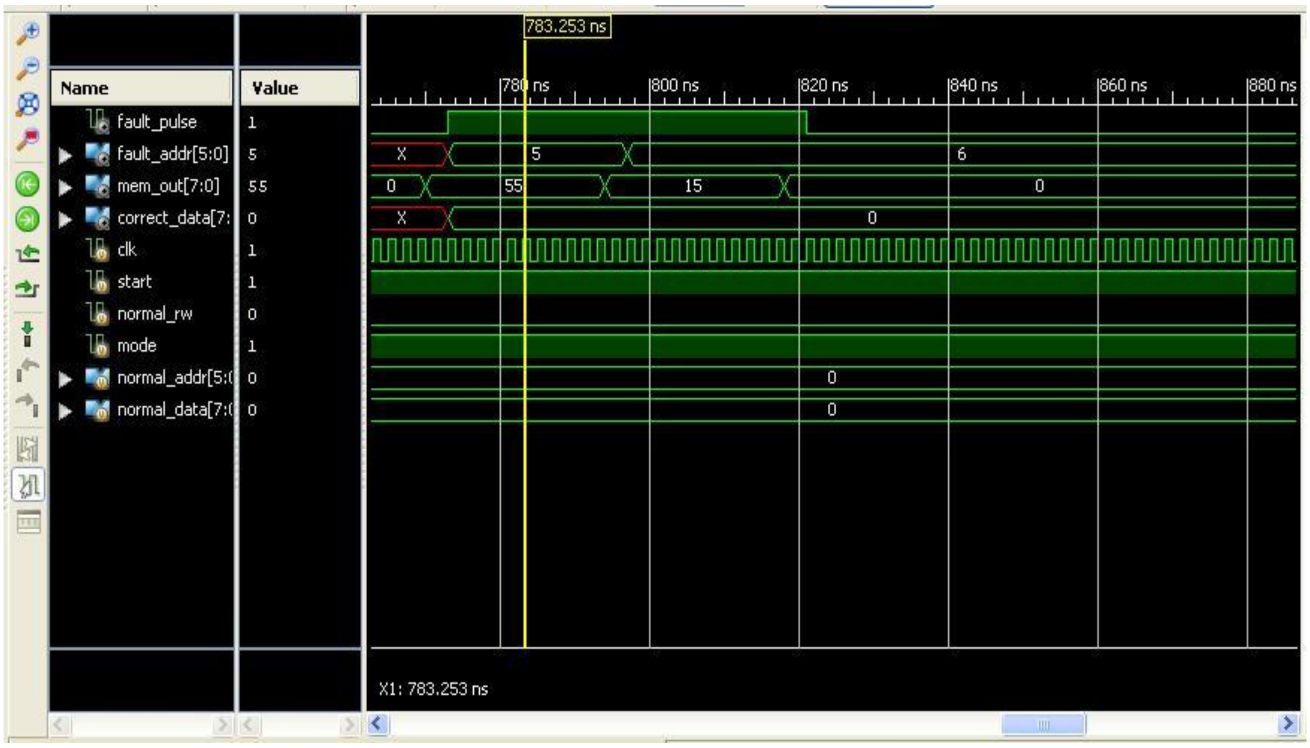

Figure 5. MBIST controller in the test mode

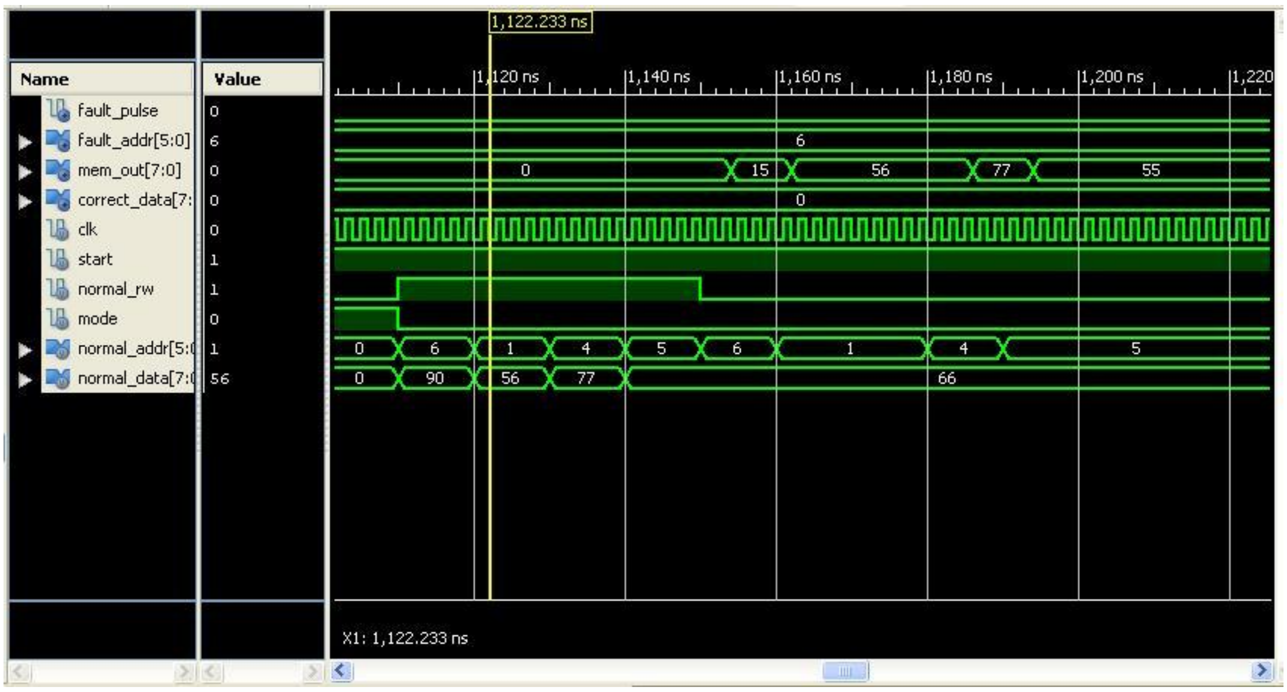

Figure 6. MBIST in normal mode

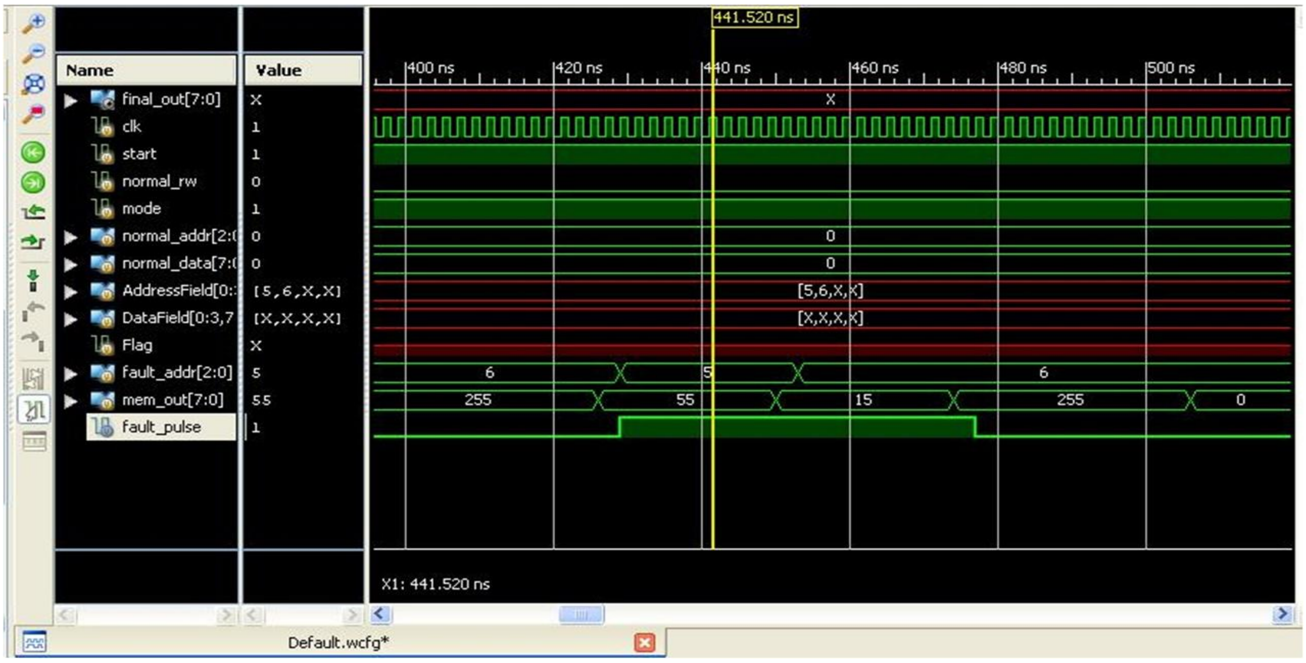

Figure 7. MBISR in the test and repair mode

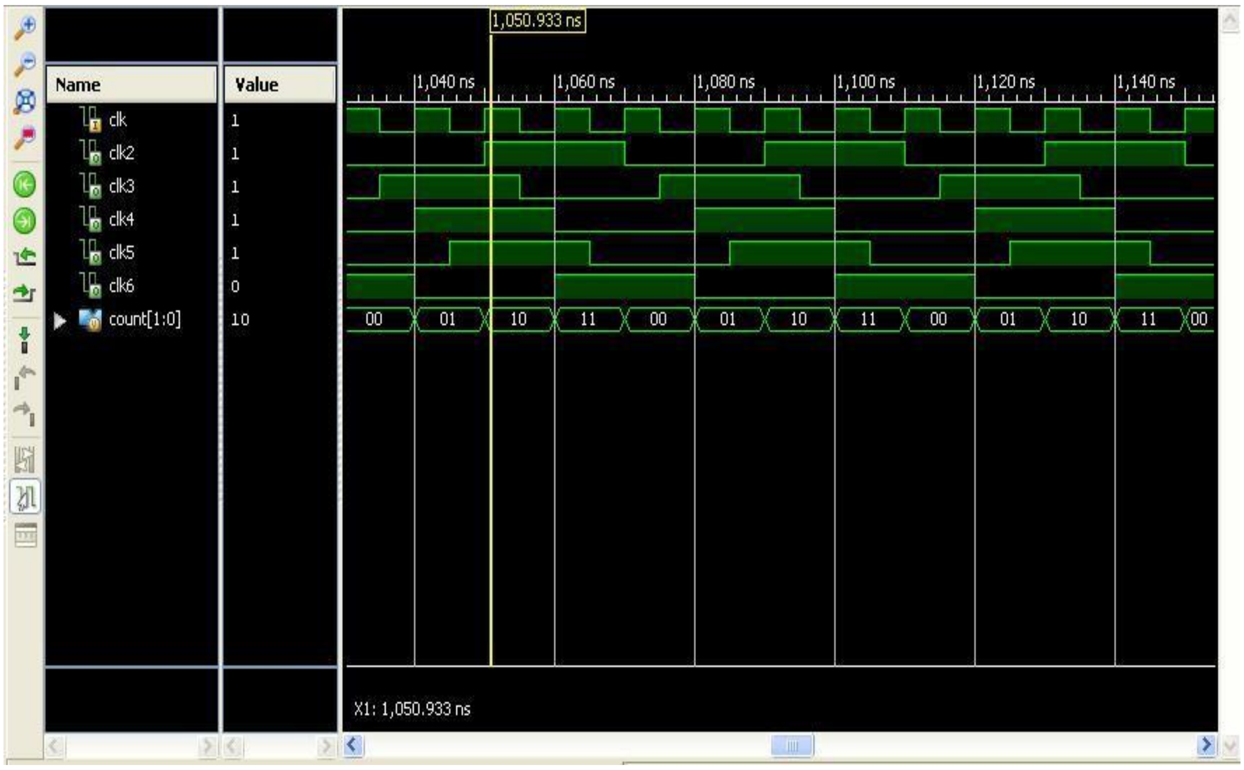

Figure 8. MBISR in the normal mode.

8. REFERENCES

[1] R. Rajsuman, "Design and test of large embedded memories: An overview," IEEE Des. Test Comput., vol. 18, no. 3, pp. 16–27, May2001.
[2] International SEMATECH," International Technology Roadmap for Semiconductors (ITRS) : Edition 2001".
[3]"March SS: A Test for All Static Simple RAM Faults", by Said Hamdioui, Ad J. van de Goor, Mike Rodgers.
[4] S. Hamdioui, et. al, "Importance of Dynamic Faults for New SRAM Technologies", In IEEE Proc. Of European TestWorkshop, pp. 29-34, 2003.

[5] N. Z. Haron, S. A. M. Junos, A. S. A. Aziz, " Modeling and Simulation of Microcode Built-In Self Test Architecture for Embedded Memories", In Proc. of IEEEInternational Symposium on Communications and Information Technologies pp. 136-139,2007.
[6] R. Dean Adams, "High Performance Memory Testing: Design Principles, Fault Modeling and Self-Test", Springer US, 2003.

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

[8] 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.

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

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

[11]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.
[12] Tsu-Wei Tseng,Jin-Fu Li, "A Low-Cost Built-In Redundancy-Analysis Scheme for Word-Oriented RAMs With 2-D Redundancy," IEEE Trans. On VLSI Systems, vol. 19, no.11,November 2011.

[13] T.-W. Tseng, J.-F. Li, and C.-C. Hsu, "ReBISR: A reconfigurable built-in self-repair scheme for random access memories in SOCs,"IEEE Trans. on VLSISyst.ems, vol. 18, no. 6, pp.921–932, Jun. 2010.
[14] T.-W. Tseng and J.-F. Li, "A shared parallel built-in self-repair schemefor random access memories in SOCs," presented at the Int. Test Conf.(ITC), Santa Clara, CA, Oct. 2008.