Design and Simulation of Microcode based Built-In Self Repair for Embedded Memories to Enhance Fault Coverage

1Mohammed Arif, 2Ankita Gupta
1Assistant Professor, 2Student M.Tech (IV) Semester
1Electronics and Communication, 2Embedded System and VLSI Design
Gyan Ganga Institute of Technology and Sciences, Jabalpur
Arif.mtech11@gmail.org
Ankita.at.ggits@gmail.com

***Abstract*: In recent time, it has been surveyed that the System-on-Chips (SoCs) are moving from logic dominant chips to memory dominant chips in order to deal with todays and future application requirements. These changing of technologies give rise to new defects and new fault models, for which there is a need for new algorithms which are defined to detect and eliminate these new defects. These new fault models are used to develop new high coverage test and diagnostic algorithms. The higher the fault detection and localization coverage, the higher the repair efficiency, hence higher will be the obtained yield. Memory repair becomes important, since only detecting the faults is not only sufficient for SoCs. March BLC algorithm is a newly developed test algorithm which deals with detecting some recently developed static and dynamic fault models. In order to use this algorithm new microcode BIST architecture is presented here which is capable of employing these new algorithms. A word-oriented BISR array is used to detect the faulty memory locations and repair those faulty memory locations by routing them to the redundant array.**

***Keywords* - Built-In Self Repair (BISR), Built-In Self Test (BIST), Defect-Per Million (DPM), Memory BuiltIn Self Repair (MBISR) , Memory Built-in Self Test (MBIST) and Microcoded MBIST**.

I. INTRODUCTION

In the survey, it has been found, that as the area for embedded memory is increasing exponentially, problem of faults is also growing exponentially. Also, as the memories grow in size and speed, the bit lines, word lines and address decoder pre-select lines will have high parasitic capacitance in addition to a high load. This increases their sensitivity for delay and timing related faults. Thus, newer test algorithms are developed for detecting these new faults. The new March algorithms have more number of operations than the existing March algorithms.

Many types of memories, with different sizes are included in a SoC. These memories are subject to aggressive design rules causing them more prone to manufacturing defects. Testing has to done to detect the faults. Testing is no longer enough for embedded memories in the SoC era. Repair mechanism BISR is also required to repair these memories.

According to the 2001 ITRS, embedded memory area of System- on- Chips (SoCs) is increasing. The dominating logic (about 64% in 1999) is changing to dominating memory (approaching 90% by 2011 and 94% by 2014) as shown in Fig.1 below.

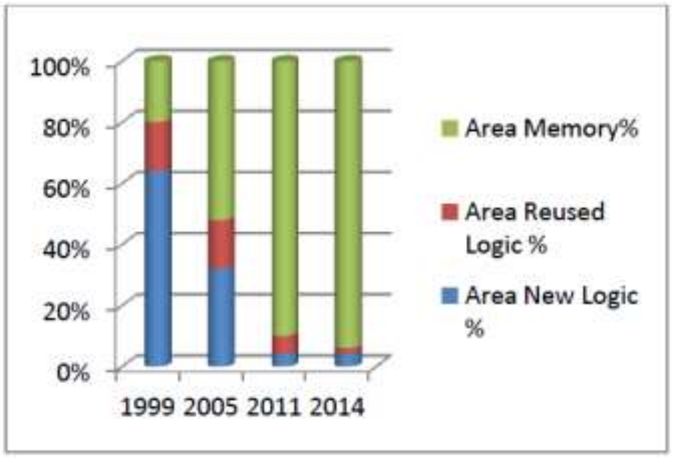


Fig. 1: Embedded memory area increasing graph over years

The new trends in memory testing will be implemented by,

Fault modeling: The fault models should be established to deal with the new defects introduced by current and future technologies.

Test algorithms: Optimal test algorithms guarantee high fault detection coverage for the new memory technologies and reduce the DPM level.

BIST: It is utilized for fast testing to distinguish shortcomings in installed recollections. Memory BIST is a standout amongst the most practical and broadly utilized answers for memory testing for the reasons like, no necessity of outside test gear, for example, Automatic test hardware (ATE), diminished improvement endeavors, and test can keep running at circuit velocity to yield a more sensible test time.

BISR: Combining BIST with proficient and minimal effort repair plans to enhance the yield and framework unwavering quality. Walk BLC test calculation can manage distinguishing some as of late created static and element deficiency models.
The interface of repair array with BIST controller and Memory under test is shown in Fig. 2.

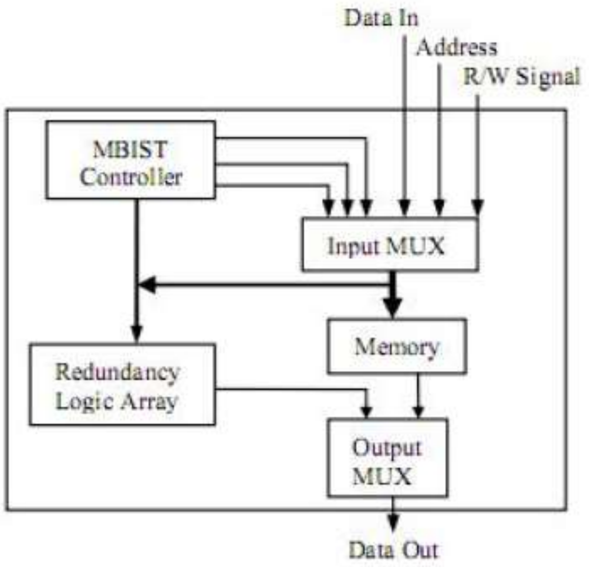


Fig 2: Block diagram of BISR

II. ARCHITECTURE OF BISR

The design of BISR which have been created to actualize before tests like March C-will be unable to effectively execute these more up to date test calculations. The reason is that the greater part of the recently created calculations have up to seven or eight number of test operations per test component.

March calculations can be effectively actualized and connected utilizing this design. This has been shown in the present work by executing March BLC calculation. The same equipment has likewise been utilized to execute other new March calculations. This requires simply changing the Instruction storage unit, or the guideline codes and grouping inside the direction storage unit. The guideline storage unit is utilized to store foreordained test design.

The block diagram of the BIST controller architecture together with fault diagnosis interface through input MUX shown in above Fig.3.

It consists of the Instruction Pointer, Instruction storage, Instruction register, AddrGen, DataGen, DataGen, RWControl, Input multiplexer, memory, redundant logic array, output multiplexer, fault diagnosis, SMC controller.

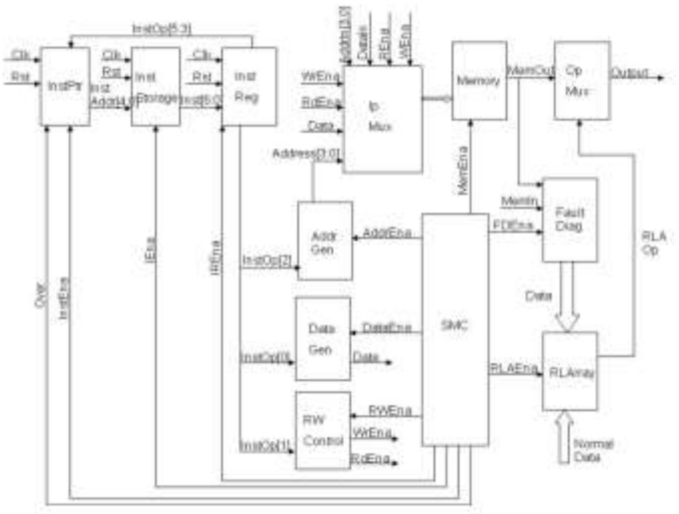


Fig.3: Architecture of Built in Self-Test and Repair

Instruction Pointer is utilized to point the direction we need to get from the direction storage. It works for each rising edge of the Clk relying upon the empowering signs and Rst values. In the event that Rst is dynamic low InstAddr is reset to zero.

Instruction Storage is utilized to store the directions. 22 directions are utilized to discover the faults in the memory. These guidelines are put away in the direction register. On the off chance that Rst is dynamic high, for each rising edge of the Clk if InstEna is dynamic high one guideline will be gotten from the memory from the area direction address pointed by guideline pointer.

Instruction Register is utilized to store the guideline which is turning out from the direction storage. The guideline is decoded and given as data to the required modules. It is a 7-bit register. In the event that Rst is dynamic low guideline register worth is reset to zero .If Rst is dynamic high and for each rising edge of the Clk, if IREna is dynamic high direction is put away in the guideline enroll and decoded.

AddrGen is utilized to produce the location. In the event that Rst is dynamic low, deliver will be reset to zero, generally for each rising edge of the Clk, if AddrEna is dynamic high address will be increased or decremented as per the data signals.

DataGen is utilized to produce the information, which is given as data to the memory. In the event that Rst is dynamic low, information will be reset to zero, generally for each rising edge of the Clk, if DataEna is dynamic high information will accord to the data signals.

RWControl is utilized to create the RdEna and WrEna signals, which are given as inputs to the memory. On the off chance that Rst is dynamic low, then RdEna and WrEna signs will be reset to zero, generally for each rising edge of the Clk, if

RWEna is dynamic high then RdEna and WrEna signs will be set by information signals.

Input Multiplexer is used to select one group of signals depending on whether it is working in Normal/Test mode. Multiplexer output is given as input to the memory.

Memory is used as a unit under test. If MemEna, WrEna both are active high and RdEna is active low, the data is written into the memory location specified on the address signal. If MemEna, RdEna both are active high and WrEna is active low, the data is from the memory location specified on the address signal.

Fault Diagnosis is used to compare the expected data with the original data. If any change is there.

Redundant Logic Array: Redundant Logic Array acts as the redundant memory. In this we will store the memory faulty locations address and data. In normal mode it compares normal input address with the existing faulty locations; if it matches it uses redundant logic memory for read and write operations. If it doesn't match it will use the original memory for read and write operations. Output multiplexer is used to select one value from the Redundant memory and Memory depending whether it is faulty or not.

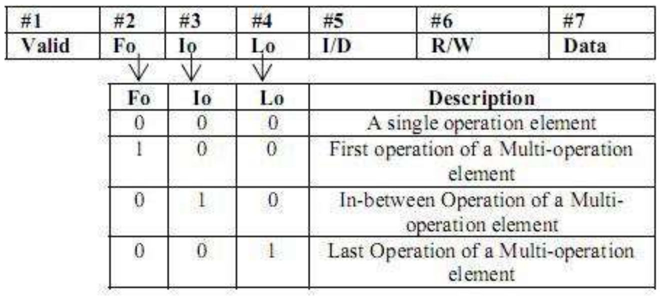
III. SPECIFICATION OF MICROCODE INSTRUCTION

March BLC test algorithms have up to six or seven operations for every March component, and subsequently a percentage of the as of late displayed and reproduced structures are insufficient to execute these test calculations, as they have been produced to make space for just up to two test operations per March component. This engineering is equipped for executing the recently created March calculations, on account of its capacity to execute calculations with boundless number of operations per March component. In this way huge numbers of the as of late created March calculations can be connected utilizing this design.

In this paper we exhibit March BLC, an advanced test that distinguishes all static shortcomings within the sight of BL coupling utilizing just the required CBs, with a test time many-sided quality of 46n. Contrasted with March m-MSS (108n), which applies all conceivable CBs, the test time is essentially decreased by more than half.

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. Examples of such tests are March BLC

March BLC = {(w0); ME0
 (r0, r0, w0, r0, w1, w1, r1); ME1
 (r1, r1, w1, r1, w0, w1); ME2
 (r1, r1, w0, w0, r0); ME3
 (r0, r0, w0, r0, w1, w1, w0); ME4
 (r0, r0, w0, w1, w1, r1); ME5
 (r1, r1, w0, w1); ME6
 (r1, r1, w0, w0, r0); ME7
 (r0, r0, w1, w1, w0)} ME8



#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 microcode direction created in this work is coded to mean one operation in a solitary micro word. Therefore a five operation March component is made up by five small scale code words. Table 1 demonstrates the 7-bit microcode MBIST Instruction word and portrayal of its different fields.
Bit #1 (=1) demonstrates a substantial microcode guideline, else, it demonstrates the end of test for BIST Controller.
Bits #2, #3 and #4 stand for first operation, in the middle operation and last operation of a multi-operation March component. A definite portrayal of how these three bits are deciphered is given below.
Bit #5 (=1) advises that the memory under test (MUT) is to be tended to in diminishing request; else it is gotten to in expanding request.
Bit #6 (=1) shows that the test design information is to be built into the MUT; else, it is recovered from the memory under test.
Bit #7(=1) implies that a byte of 1s is to be produced (composed to MUT or anticipated to be perused out from the MUT); else eight bits of every one of the zeroes are produced.

IV. WORD REDUNDANCY MBISR

The BISR system utilized here utilizes a variety of repetitive words set in parallel with the memory. These excess words are utilized as a part of spot of broken words in memory. For fruitful interfacing with as of now existing BIST arrangements as appeared in Fig. 2, The accompanying interface signs are taken from the MBIST rationale: 1) A flaw beat showing a broken area address, 2) Fault address, 3) Expected information or right information that is contrasted and the aftereffects of Memory under test.
The MBISR rationale utilized here can work as a part of two modes.

A) Mode 1: Test and Repair Mode

In this mode the info multiplexer interfaces test neckline contribution for memory under test as produced by the BIST controller hardware. As broken memory areas are recognized by the issue determination module of BIST Controller, the excess exhibit is customized. An excess word is as appeared in Fig 4.

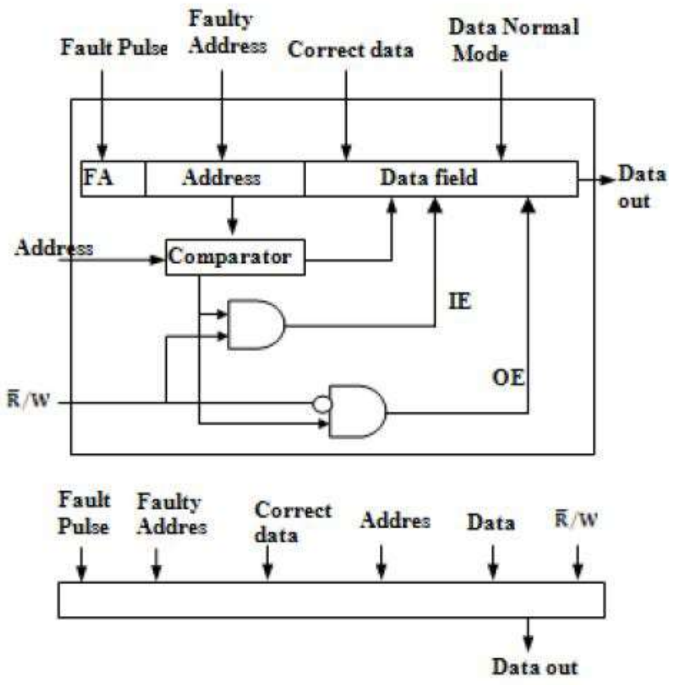


Fig.4: Redundancy Word Line

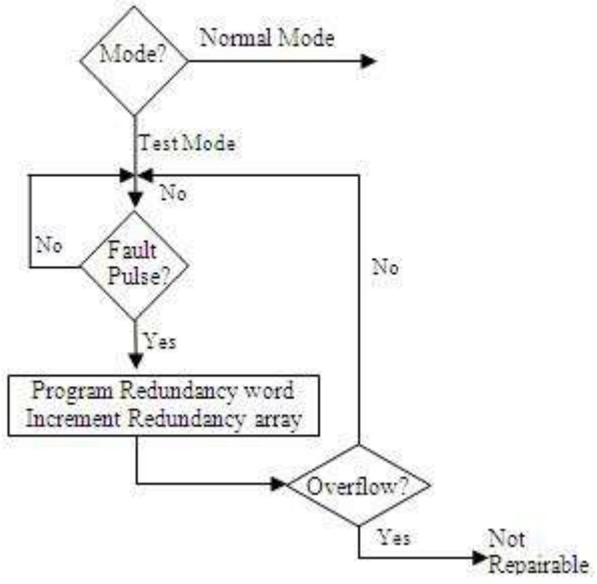


Fig.5: Flowchart of Redundancy Array

B) **Mode2*: Normal***

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 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 (= 0) of the faulty location in case of a read signal. This can be easily understood by the redundancy word.

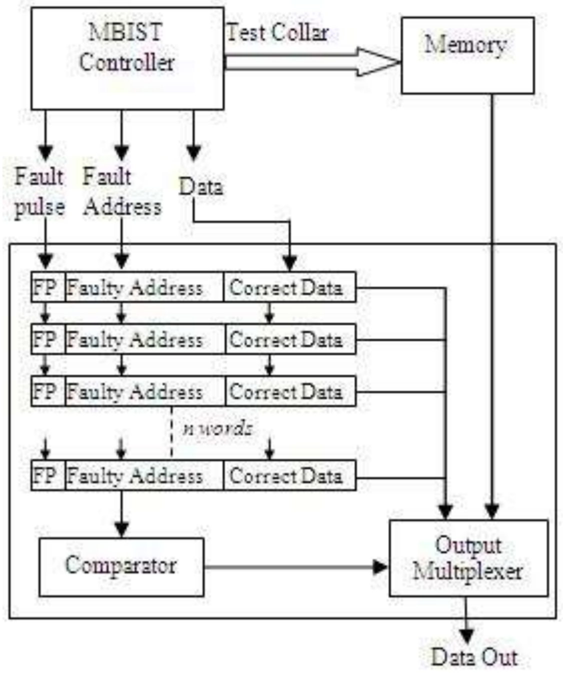


Fig.6: Repair module

The above Fig .6, shows the repair module including the redundancy array and output multiplexer and its interfacing with the existing BIST module.

V. SIMULATION RESULTS

The block diagram of top module is shown in Fig.7

Fig.7: Simulated waveform of clock generator module

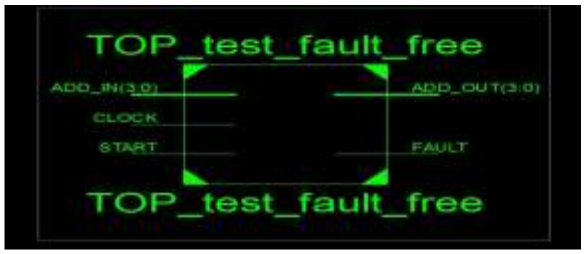


Fig.8: Block Diagram of Top Module

A) The fault free diagram of BSIR module is shown below.

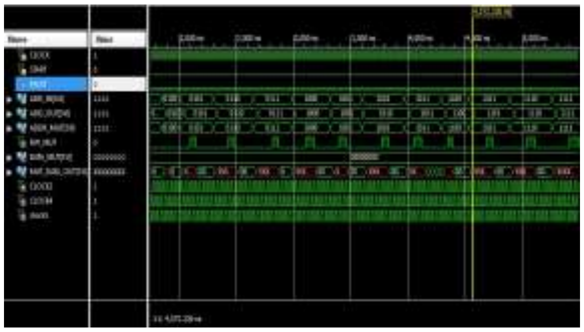


Fig.9: Simulated waveform of Fault-free SRAM

B) Faulty output for the same is show below.

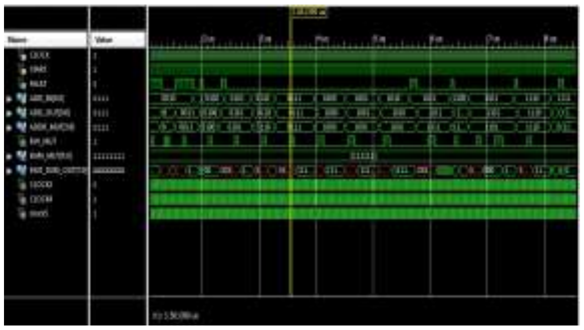


Fig.10: Simulated waveform of Faulty SRAM

C) Deceptive read fault- writing 1 at location 1011.

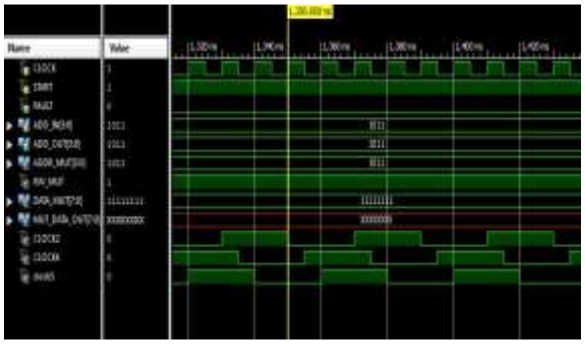


Fig. 11 Deceptive read fault

D) Reading 1 at location 1011 to verify

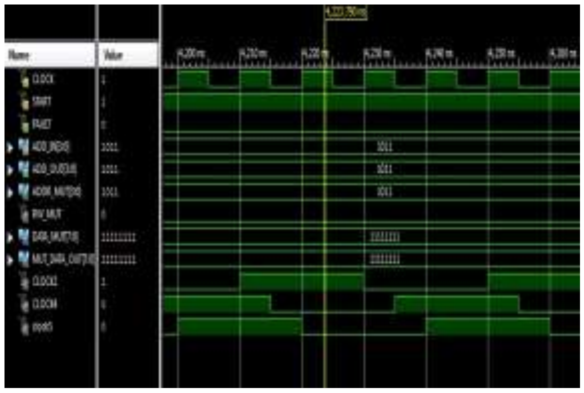


Fig.12: Waveform for reading 1 at 1011

E) Due to deceptive read fault, data at 1011 is found inverted on reading.

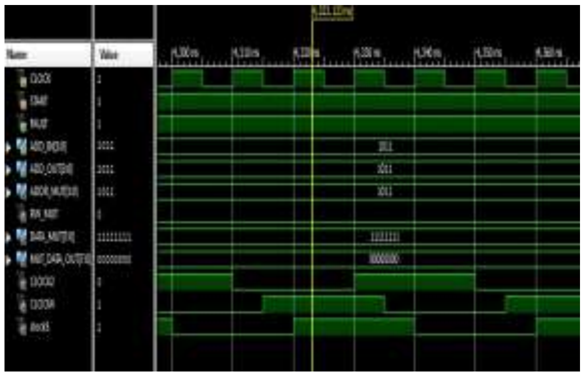


Fig.13: Waveform found faulty on reading 1 at 1011

F) Repair mechanism applied, and correct output is obtained.

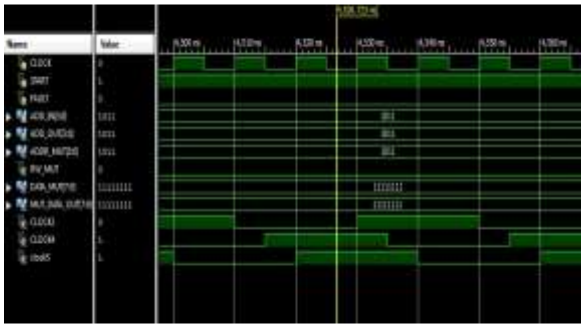


Fig.14: Waveform found fault free after repair

VI. CONCLUSION

The simulation results have demonstrated that the micro-coded BISR architecture is effectively ready to execute new test algorithms. Execution of a single test operation in one micro word guarantees that any future test algorithms with any number of test operations per test component are effectively actualized utilizing the current BISR architecture.

In addition, it gives an adaptable methodology as any new march algorithm, other than March BLC can likewise be actualized utilizing the same BIST hardware by changing the directions in the microcode storage unit, without the need to upgrade the whole hardware. The Synthesis Report, Map Report, RTL Schematics, Floor Plan Design are produced utilizing Xilinx 9.1i. The simulation results are produced and confirmed.

REFERENCES

[1]Vinod Kapse, Mohammed Arif, “Optimization of microcode Built-In Self-test By enhanced Faults Coverage for Embedded Memory”, IEEE Students’ Conference on Electrical, Electronics and Computer Science.

[2] P.Ravinder, N.Uma Rani, “Design and Implementation of Built-in-Self Test and Repair”, International Journal of Engineering Research and Applications (IJERA)

[3] D.Satheesh Chandra Kumar, G. Kiran Kumar, “Design and Simulation of Micro Based Built In Self-Test For Fault Detection and Repair in Memories”, International Conference on Electronics and Communication Engineering, 20th, May 2012, Bangalore, ISBN: 978-93-81693-29-2

[4] International SEMATECH, “International Technology Roadmap for Semiconductors (ITRS): Edition 2001”

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

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