GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

DESIGN AND IMPLEMENTATION OF BUILT IN SELF TEST USING MARCH ALGORITHM

Chand Rani[1], Neeru Jakher[2] & Sumit Dalal[3]
[1,2,3]Electronics & Communication Department, [1,2,3]Maharishi Dayanand University
[1]M. Tech. Scholar, [2,3]Asst. Prof., [2,3]Sat Kabir Institute of Technology & Management

ABSTRACT

In today's circuit designs Built-in Self-Test (BIST) is becoming important for the memory which is the most necessary part of the System on Chip. The March algorithm has been widely used to test memory core of System on chip (SOC). LFSRs and counters are mainly used to generate the memory addresses, which can be serially applied to the memory cores under test. In this paper Address counters and Data generators (i.e. parts of the MBIST) are designed. These implemented in Hardware Description Language (HDL), and the area and power analyzed for each case.

Keywords: *BIST, Low power, Address counter, Test Pattern Generators.*

I. INTRODUCTION

Memories are the most universal component today. Almost all system chips contain some type of embedded memory, such as ROM, SRAM, DRAM, and flash memory. In the realm of PC, Alpha 21264, with the advent of deep-submicron VLSI technology, the memory density and capacity is growing. The clock frequency is never higher. 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. Built-in self-test (BIST) has been proven to be one of the most cost-effective and widely used solutions for memory testing for the following reasons:

(1) No external test equipment.
(2) Reduced development efforts.
(3) On-chip response analysis.
(4) Test can be on-line or off-line.
(5) Adaptability to engineering changes.
(6) Easier burn-in support.

II. METHODOLOGY

We have proposed a structured design methodology to construct FSM-based programmable memory BIST. The proposed BIST can be programmed on-line, with a "macro command", to select a test algorithm from a memory BIST. In general, there are a variety of heterogeneous memory modules in SOC, and it is not possible to test all of them with a single algorithm. It achieves a good flexibility with smaller circuit size compared with previous methods.

III. ALGORITHM

A March-based test algorithm is a finite sequence of March elements. A March Element is specified by an address order and a number of reads and writes. Since March-based tests are all simple and possess good fault coverage, they are the dominant test algorithms implemented in most modern memory BIST.

In order to verify whether a given memory cell is good, it is necessary to conduct a sequence of write and read operations to the cell. The actual number of read/write operations and the order of the operations depend on the

target fault model. Most commonly used memory test algorithms are March tests, in which there are finite sequences of March elements. A March element is a finite sequence of read (r) or writes (w) operations applied to a cell in memory before processing the next cell. The address of the next cell can be in either ascending or descending address order.

IV. PROPOSED PROGRAMMABLE BIST ARCHITECTURE

The proposed programmable MBIST architecture consists of two parts, as shown in figure 1.1. An Algorithm Generator has algorithm selection table and produces the March elements according to the selected test algorithm in correct sequence.

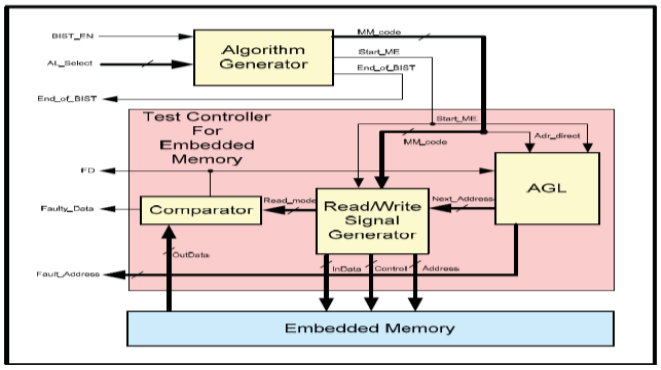


Figure 1.1: Proposed Programmable BIST Architecture [2]

A Test Controller generates read/write operations for embedded memory, address to access memory. The Comparator of the Test Controller compares a read data from memory and an original data from March element. It determines whether a current memory cell is good or bad.

Each part has a finite-state-machine (FSM). In figure 1.1, a FSM of the Algorithm Generator is operated by signal 'BIST_EN'. When an algorithm selection code is set in the Algorithm Generator, the Algorithm Generator sent a March element code to the Test Controller. Then the Generator waits a signal 'End_MM' meaning done of March element. The Algorithm Generator sent a next March element code after receiving a signal 'End_MM' from the Test Controller. If all of March element codes of an algorithm are applied, FSM generates signal 'End_of_BIST'. FSM of Algorithm Generator in the Test Controller, an address to access memory is made according to an address order. It is also generated that memory access signal for read/write operations, in figure 1.1. The Address Generator uses the most significant bit of the March element code. It generates a new address, when the Read/Write Signal Generator completes an operation of the March element code.

V. WORKING TECHNOLOGY

In the first block " Algorithm Generator" has been shown in which all 8 Algorithm are kept in ideal state this algorithm will remain in ideal state unless BIST_EN signal is made equal to "1" (or it can be understood as until BIST _EN signal is made equal to "1") the BIST operation would not be started among this 8 algorithm one of the algorithm will be selected depend on the selection line of 8:1 MUX then selected algorithm is passed to "Algorithm Decode" read/write signal from this block this signal is transferred to the Embedded RAM to define weather read operation or write operation will be performed on to RAM Address up/down signal is passed to the Address Generator to define weather Read /Write operation is performed in Up addressing mode or Down addressing mode.

ISSN 2348 – 8034
Impact Factor- 5.070

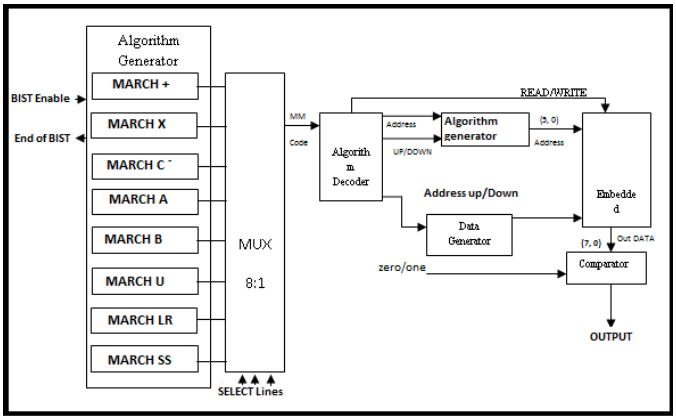
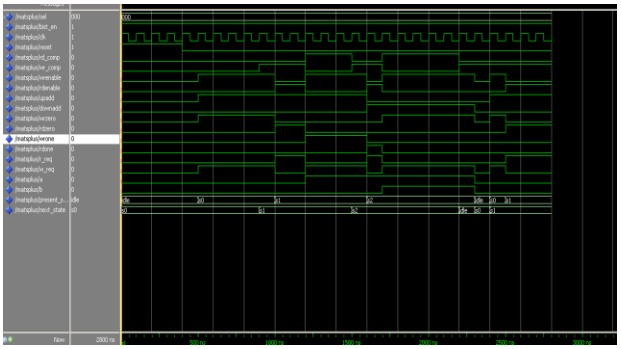


Figure 1.2 Working Technology

It is decided by selected algorithm operation then third signal Zero /One is Passed to the Data Generator to define that read/Write operation is performed on "0" bit or "1" bit hence according to the state of selected algorithm Read/write operation is performed on to the RAM whenever Read operations performed then data comes out of RAM and that data is transferred to the Comparator In comparator there is another input port (that is test data) in which test input is given as same as that of data written on to Embedded RAM , In comparator it will checked that both Input are same or not ,if there is some fault on (like stuck at fault ,or coupling fault), then the output data coming out of Embedded Memory ,is different from the input given in to the comparator so it will show fault by making " fault detect" signal "equal to "1" ,if there is no fault in memory then "fault detect" will be equal to the "0" On other hand if Write is being performed then comparator will be ideal state.

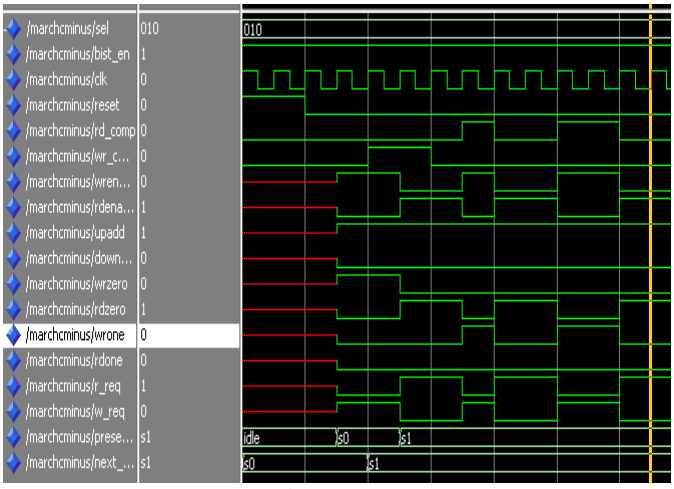
VI. IMPLEMENTATIONS & SIMULATIONS



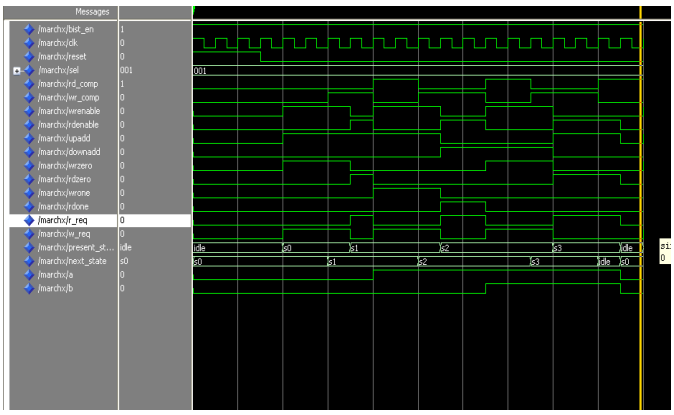
Implementation of Meta plus Algorithm

Here we are showing the modelsim waveform of the MATS + algorithm the first operation "write zero (w0), which was to be performed and as shown in wave form and in second state read 0 operation was to be performed, as defined by the mats+ algorithm.

Similarly the different waveforms shown below represent the respective algorithms:

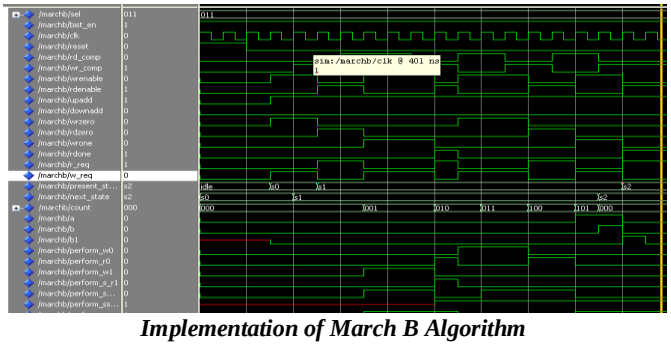


Implementation of March C Minus Algorithm

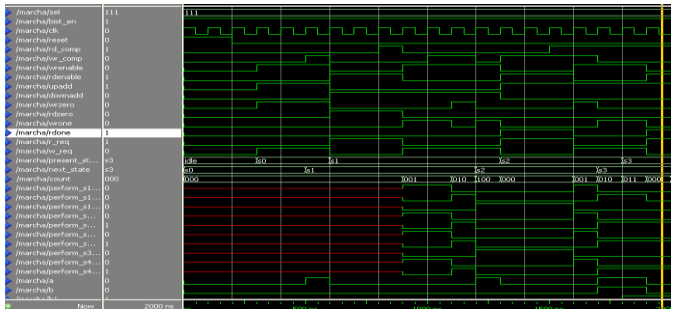


Implementation of March X Algorithm

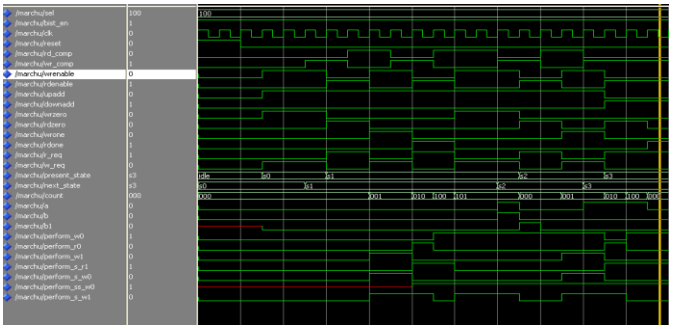
In this simulation select line is 000 is given to select the Mats+ algorithm in which S1 state is showing the read 0 operation during which output from RAM is transferred to comparator in test data to comparator is given as "1" hence stuck at 0 fault is being shown in this simulation In this simulation waveform of March X has been shown where operation is being performed according to algorithm.



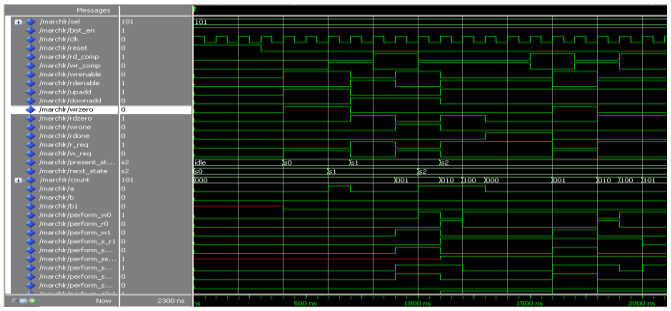
Implementation of March B Algorithm



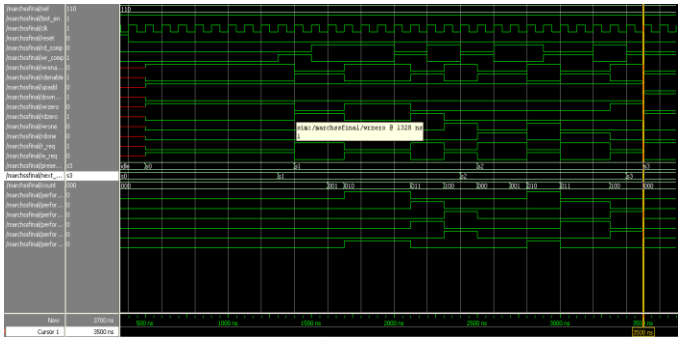
Implementation of March A Algorithm



Implementation of March U Algorithm



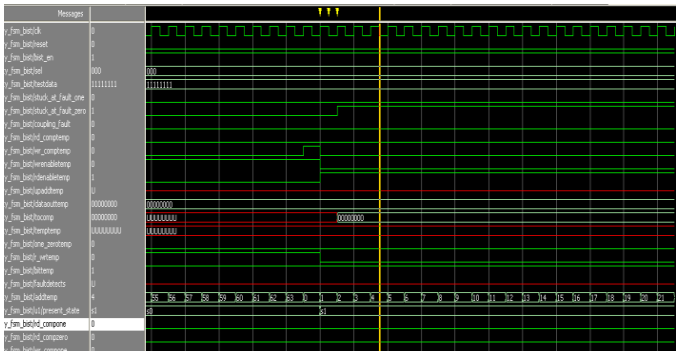
Implementation of March LR Algorithm



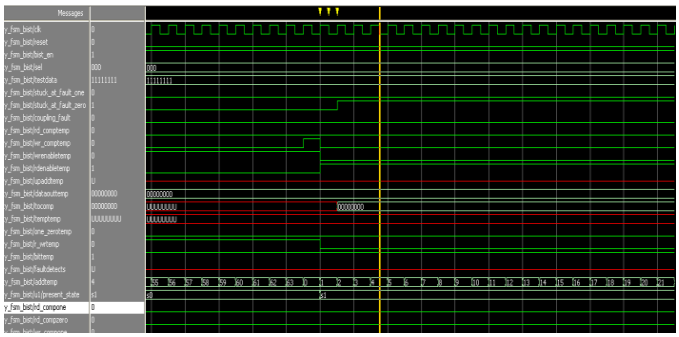
Implementation of March SS Algorithm

Here we are showing the modelsim waveform of the Marchss algorithm it is the algorithm were 6 states are there the first operation "write zero (w0) Is what which was to be performed and as shown in wave form and in second state read 0 operation was to be performed, as defined by the March algorithm and thus remaining operation of this algorithm is performed and shown in this waveform.

ISSN 2348 – 8034
Impact Factor- 5.070

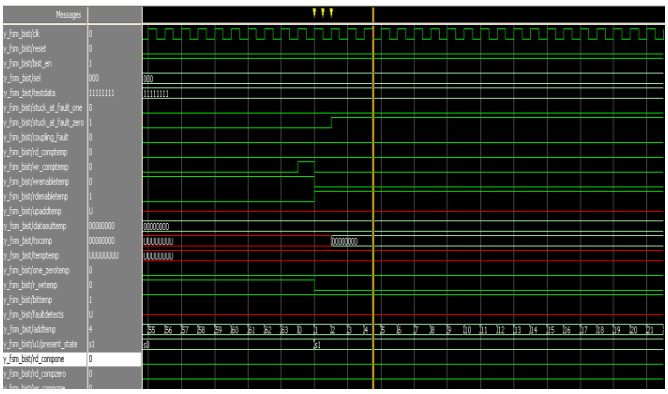


Simulation of stuck at fault



Final Implementation

In this simulation select line is 000 is given to select the Mats+ algorithm in which S1 state is showing the read 0 operation during which output from RAM is transferred to comparator in test data to comparator is given as "1" hence stuck at 0 fault is being shown in this simulation



In this simulation coupling fault is being shown as if test data is given as 01010101 and output from RAM is 00000000 coupling fault is fault which check two consecutive bits

ISSN 2348 – 8034
Impact Factor- 5.070

VII. COMPARISON (FULL COVERAGE)

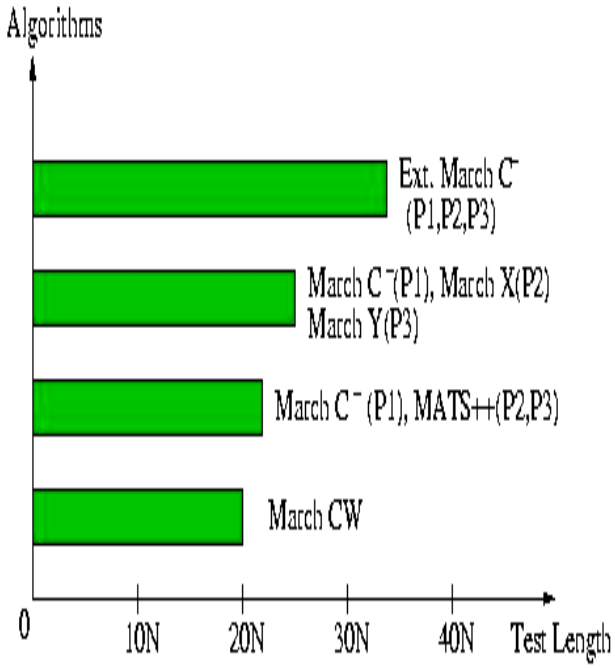


Figure 1.3: Comparison between different algorithms

Our synthesis result (table1.1) shows that our Gate Count for March algorithm is much lower as compared to previous results like Mats+, in our work, the gate count is 216, and for March X the gate count is 241, ant that for March C Minus is 281 but for March B we have not been able to reduce the gate count which is 1, 215 for our scheme hence our gate count is a just a bit higher.

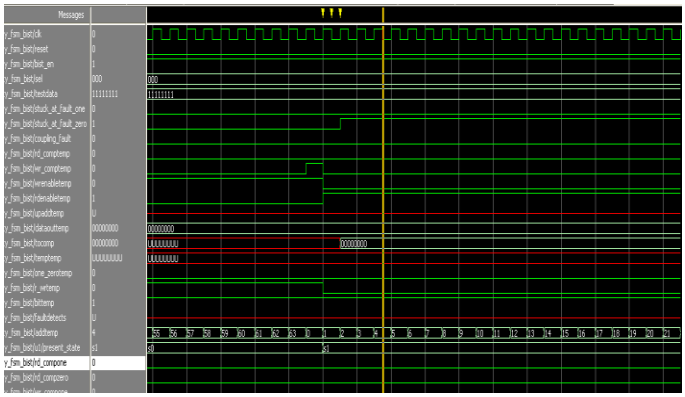
Table 1.1: Result

Algorithm	Gate Count of Previous method	Gate Count of Proposed method
MATS+	730	216
March X	768	241
March C-	762	281
March B	1,038	1,215
March LR	NI	905
March U	NI	885
March SS	NI	651

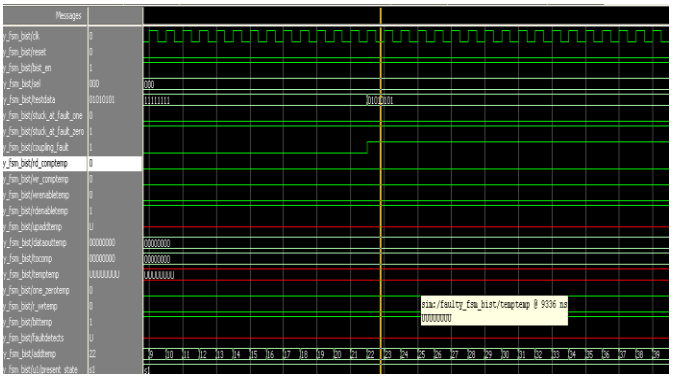
VIII. EXPECTATIONS & ACHIEVEMENT

In our project the goal is to implement some March algorithm of Memory BIST having lower area as compared to previous scheme (described in table) in which we are showing the gate count comparison between the previous and the our proposed scheme (after diagram). Our first objective is to find faults in Device under Test (Embedded RAM

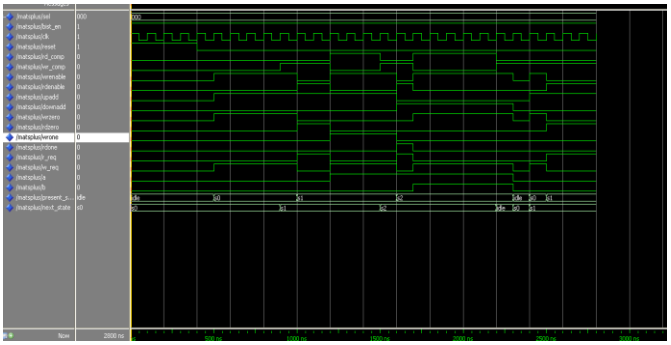
in our scheme). So we have proposed our design to detect 3 faults like stuck at '0' fault stuck at '1' fault coupling Fault. The waveform that we have obtained from modelsim simulation is showing

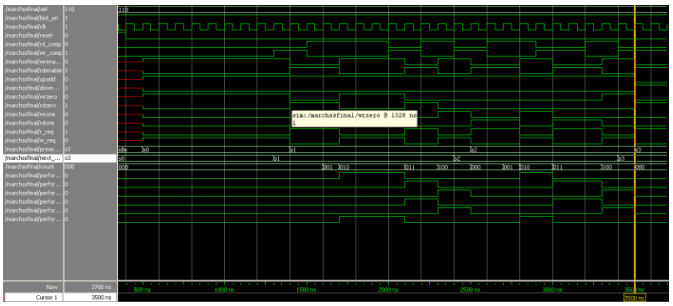


In this simulation sel line is 000 is given to select the Mats+ algorithm in which S1 state is showing the read 0 operation during which output from RAM is transferred to comparator in test data to comparator is given as "1" hence stuck at 0 fault is being shown in this simulation



In this simulation coupling fault is being shown as if test data is given as 01010101 and output from RAM is 00000000 coupling fault is fault which check two consecutive bits. Every algorithm has different states. We have drawn state diagram of every algorithm according to the operation which is needed to be performed for corresponding state. Our resultant waveform is showing that operation is being performed. For example in mats plus algorithm objective is to perform {↕(w0); ↕ (r0,w1); ↕(r1,w0);}. In the simulated waveform of Mats plus idle state was the first state which when reset is made equal to 1 goes to the S0 state. When in the first state when wr o operation is completed then state goes to the third state S1 read 0 and write 1 operation was needed to performed so when wr 1 is equal to 1 which when made equal to one machine moves to third state when read 1 operation is to be performed.





In above mentioned wave form March SS modelsim simulated wave has been shown, which is maintained in the idle state unless BIST_En signal is made equal to 1. Hence all remaining operation is also performed as expected earlier.

IX. SUMMARY & CONCLUSIONS

Proposed scheme is able to test the memory for limited number of faults only, like stuck at '0', fault stuck at '1', fault coupling Fault, which is the major limitation of our project. As in any BIST, the fault coverage is very important which is able to test the memory for maximum fault, would be more efficient which can be improved by making our BIST sensitize for other faults; Transient faults, Bridging Faults. Hence fault coverage of BIST can be improved further.

Another constraints in our scheme is that; byte input is given as test data to the comparator and byte coming from memory to comparator, so our BIST is sensitive for byte change only so if suppose byte coming from memory is 00000000 and test data provided to comparator is 11111111 then fault will appeared but if any change in single bit occur then fault could not be displayed hence our design is not sensitive for change in single bit only so BIST can be made sensitize to change in single bit.

The next constraints can be improved is that our design is just able to detect the fault in Device under Test but it is not able to debug (repair) the fault so BIST can be improved in that aspect. So in future, BIST would be used for not only testing of faults in memory but also it would be used for debugging of memories as well. Hence this future enhancement can be done. As we tried to make low area FSM BIST but in one of the algorithm (March b algorithm) greater area is required but in remaining algorithm we have been able to keep the area low.

REFERENCES

1. *R. K. Sharma Aditi Sood "Modeling and Simulation of Microcode-based Built-In Self Test for Multi- Operation Memory Test Algorithms" IJCSI International Journal of Computer Science Issues Published in Volume 7, Issue 3, No 2, pp 36-40, May 2010.*
2. *National Programme on Technology Enhanced Learning (NPTEL), Module 11: Built in Self-test (BIST), online available at: http://nptel.iitm.ac.in/courses/106103016/30.*

3. *Zainalabedin Navabi Digital System Test and Testable Design: Using HDL Models and Architectures Springer New York Dordrecht Heidelberg London; 1st Edition. (December 20, 2010).*