Modeling and Simulation Experiment on a Built-In Self Test for Memory Fault Detection in SRAM

G.Narahari[#1], R Venkata Sai Kiran Epi[*2], Tarannum Sultana [#3]

[#1]*Assistant Professor, Department of Electronics and Communication Engineering, Sri Sai Jyothi Engineering College, Gandipet, Hyderabad-75, (A.P.), India.*

[*2]*P.G. Student, M.Tech. (VLSI), Department of Electronics and Communication Engineering, Sri Sai Jyothi Engineering College, Gandipet, Hyderabad-75, (A. P.), India.*

[#3]*Assistant Professor, Department of Electronics and Communication Engineering, Sri Sai Jyothi Engineering College, Gandipet, Hyderabad-75, (A.P.), India.*

Abstract:

The tremendous increase in memory area on chip is in turn increasing memory density and causing problems (i.e. faults) in Memories to detect these fault's we require tests with high fault coverage and low cost. The faults may be of different types e.g. static faults and dynamic faults so many has proposed different Algorithms to detect these kind of faults (e.g. March algorithms) .The proposed paper deals with Static and Dynamic Faults, these are established and evaluated for static random access memories (SRAM), by using newly developed micro-coded MBIST architecture which can be used to employ these new test algorithm which consists of 14 instructions and have much less number of operations than the Previous Existing March algorithms (March SS, LR e.t.c),which increases the fault coverage and reduces the time for Detecting Faults. MBISR architecture function's in two modes of operations. Mode 1: Normal mode and Mode 2: Test and repair mode. A Built-in-Self repair Methodology is used to repair the faulty locations indicated by the MBIST controller. The architecture is designed by employing Our proposed method of testing faults is simulated by using Xilinx 9.1ISE versions and synthesized by using Xilinx 9.1 ISE version.

Keyword - Built-In Self Test (BIST), Built-in Self Repair (BISR), Memory Built-in Self Test (MBIST), Reliability, Static and Dynamic faults.

1. INTRODUCTION:

During the fabrication of a chip, some errors may occur which may induce the faults in the circuit devices. This results in the inaccurate functioning of the chip. Hence the chips need to be tested for the presence of any faults else it may result in the inaccurate functioning of the chip. Since the number of circuits on the chip is increasing exponentially, testing of such multimillion transistor chip causes a serious problem, and this introduces the concept of *Design For Testability (DFT)*. Design for testability (DFT) refers to including test considerations into the design specifications. DFT includes using design rules that forbid the use of certain hard-to-test circuit forms and/or require using inherently testable forms. It attempts to ensure that internal nodes are sufficiently controllable and observable. Specific DFT techniques include providing parallel test modes to test multiple arrays simultaneously, *Built-In Self-Test (BIST), Built-In Self-Repair (BISR*).

In recent years, embedded memories are the fastest growing segment of SoCs. They therefore have major impact on the overall Defects per Million (DPM). According to 2001 International Technology Roadmap for Semiconductors (ITRS 2001), today's SoCs are moving from logic dominant chips to memory dominant chips, since future application will require lot of memory. The memory share on the chip is expected to be about 94% in 2014.

Hence cost of testing memories increases rapidly with every new generation of memory Chips. Therefore Precise fault modeling and efficient test design is essential to reduce the cost and increase the test time to find faults. Now-a-days embedded memories are harder to test using ATE, and the defective cells detected by the BIST circuit are replaced by the cells of the spare SRAM. The built-in self diagnosis method presented for repairable SRAMs uses a reduced-instruction-set processor to determine a repair solution.

Reliability is one of the main considerations in any circuit design. It involves a correct and predictable behavior of the circuit according to design specifications over a sufficiently long period of time. To achieve this goal, the logic-circuit design is aimed at an error-free circuit operation. Hence, when a fault occurs anyway, one must be able to detect the presence of the fault and, if desired to pinpoint its location. This task is accomplished by testing the circuit.

Digital circuit manufacturers are well aware of the need to incorporate testability features early in the design stage, or otherwise they have to incur higher testing costs, subsequently. However, recent studies have shown that the cost of testing and fault finding, at system and field level, is increasing exponentially. Thus, if a fault can be detected at chip or board level, then significantly larger costs per fault can be avoided. This is the prime reason that attention has now focused on providing testability at chip, module or even at board level.

Any test methodology usually consists of

(i) A test strategy for generating the test-stimuli,
(ii) A strategy for evaluating output responses, and
(iii) Implementation mechanisms to realize the appropriate strategies in test-generation and response evaluation.

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

BIST: It is used at high speed testing for detect the faults in embedded memory. This is 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.

The design proposes a word oriented memory Built-in Self-Repair methodology (BISR) that targets on embedded SRAM and does not rely on spare rows and columns. It uses BIST logic to identify faulty words and redundancy logic to store faulty addresses and data immediately after its detection during test; wrapper logic to replace defect words.

II.MICROCODE BISR CONTROLLER

The Architecture of the Micro-code based built in self test and repair is shown In the Fig.1 below, It consists of the Instruction Pointer, Instruction storage, Instruction register, AddrGen, DataGen , R/W Control, Input multiplexer, Reliability block, memory, redundant logic array, output multiplexer, Fault diagnosis, SMC controller**.**

Instruction Pointer:

It points to the next Micro word instruction that has to fetch from the instruction storage and applied to memory under test. It works for every rising edge of the Clk depending on the enabling signals and Rst values. If Rst is active low InstAddr is reset to zero.

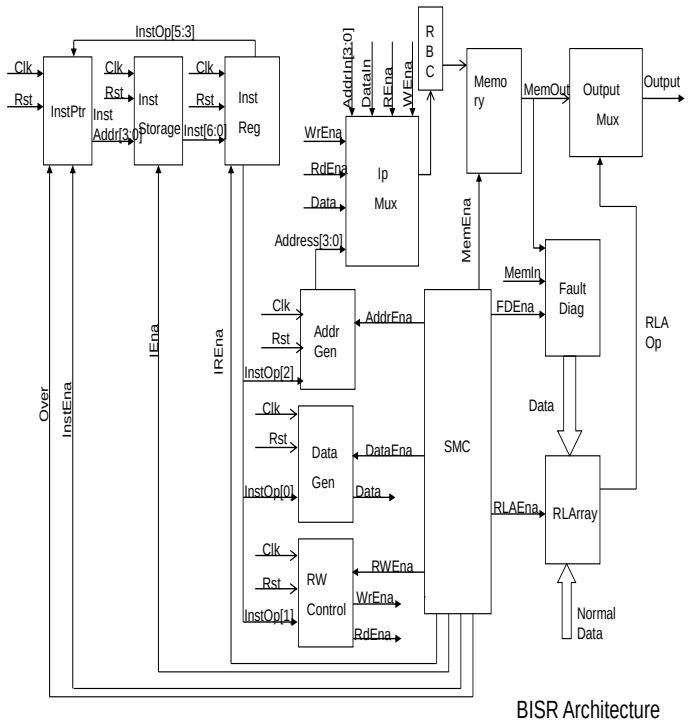

Fig.1: Architecture of Micro-coded BISR.

Instruction Storage: It is used to store the instructions i.e. 14 instructions are used to find the faults in the memory. These instructions are stored in the instruction register. If Rst is active high, for every

rising edge of the Clk if InstEna is active high one instruction will be fetched from the memory from the location instruction address pointed by instruction pointer.

Instruction Register: It holds the instruction pointed at by instruction pointer. The instruction is decoded and given as input to the required modules. It is a 7-bit register. If Rst is active low instruction register value is reset to zero .If Rst is active high and for every rising edge of the Clk, if IREna is active high instruction is stored in the instruction register and decoded.

Address Generator: It is used to generate the address. If Rst is active low, address will be reset to zero, otherwise for every rising edge of the Clk, if AddrEna is active high address will be incremented or decremented according to the input signals.

Data Generator: It is used to generate the data, which is given as input to the memory. If Rst is active low, data will be reset to zero, otherwise for every rising edge of the Clk, if DataEna is active high data will be according to the input signals.

RW Control: It is used to generate the RdEna and WrEna signals based on Micro-code bits, which are given as inputs to the memory. If Rst is active low, then RdEna and WrEna signals will be reset to zero, otherwise for every rising edge of the Clk, if RWEna is active high then RdEna and WrEna signals will be set according to the input signals.

Input Multiplexer: It gives the input to memory by considering 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. In case of Normal mode the memory responds to the external address, data and read/write signals. Multiplexer output is given as input to the Reliability.

Reliability: It is one of the main considerations in any circuit design it involves the correct and predictable behavior of the circuit according to design specification, to achieve this goal logic circuit design should be an error free circuit. Mainly it increases the life time of SRAM.

Memory: It 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: This module works especially during test mode and its function is to compare the expected data with the original data. If any changes occurred it gives that location address and actual data as input to the Redundant Logic Array.

Redundant Logic Array: It acts as the redundant memory. In this memory faulty locations address and data will be stored. 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: It is used to select one value from the Redundant memory and Memory depending whether it is faulty or not.

III. SPECIFICATION OF MICROCODE INSTRUCTION

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 newly developed algorithm. The same hardware has also 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.

The microcode is nothing but a binary code which 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, 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 micro word. Thus a three operation March element is made up by five micro-code words. The format of 7-bit microcode MBIST instruction word is as shown in Table 1. Its fields are described as follows:

1) Bit #1 indicates whether it is a valid microcode instruction or not, if it is equal to 1 it considers as an

Fo	Io	Lo	Description
1	1	1	A Single Operation Element.
0	1	1	First Operation Of a Multi-operation Element.
1	0	1	In –Between Operation Of a Multi- Operation Element.
1	1	0	Last Operation Of a Multi-Operation Element.

Valid Instruction otherwise it indicates the end of test for BIST Controller.

2) Bits #2 is used to Specify first operation of a multi- operation element as shown in Table 1.

3) Bits #3 is used to Specify In-between operation of a multi- operation element as shown in Table 1.

4) Bits #4 is used to Specify Last operation of a multi- operation element as shown in Table 1.

5) Bit #5 is used to Specify whether the address should be increasing or decreasing order if it is equal to 1 it notifies that the memory under test (MUT) is to be addressed in increasing order; else it is accessed in decreasing order.

6) Bit #6 is used to Specify which operation i.e. read or write operation to performed. If it is equal to 1 indicates that the test pattern data is retrieved from the memory under test; else, it is to be written into the MUT.

7) Bit #7 is used to displays the data that is applied, if it is equal to 1 it signifies that a byte of 1's is to be generated (written to MUT or expected to be read out from the MUT); else byte containing all zeroes are generated.

March Notation:

A Complete march test is delimited by the '{..}' bracket pair, while a March element is delimited by the '(..)' bracket pair. March elements are separated by semicolons and the operations within a March element are separated by commas. Note that all operations of a March element are performed at a certain address, before proceeding to the next address. The latter can be done in either one of the two address orders: an increasing (↑) or a decreasing (↓) address order. When the address order is not relevant, the symbol is used.

The instruction word is so designed so that it can accommodate any existing and proposed algorithm. The contents of Instruction storage unit for Newly Developed instructions are shown in Table 3.

Table 1: Format of Microcode Instruction word

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

Table 2: Proposed Instructions

{ ↕ (W0);
M0
↑ (W0, R0, R0); ↑ (W1, R1, R1);
M1 M2
↓ (W1, R1, R1); ↓ (W0, R0, R0);
M3 M4
↕ (R0)}
M5

1)The first march element M0 is a single operation element, which writes zero to all memory cells in any order, whereas the 2)second march element M1 is a multi-operation element, which consists of Three operations: i) W0, ii) R0, iii) R0. MUT is addressed in increasing order as each of these three operations is performed on each memory location before moving on to the next memory location.

Table 3.Proposed Algorithm

	#1 Valid	**#2 Fo**	**#3 Io**	**#4 Lo**	**#5 I/D (1/0)**	**#6 R/W (1/0)**	**#7 Data (0/1)**
M0: ↕ W0	1	1	1	1	1	0	0
M1: ↑{ W0	1	0	1	1	1	0	0
R0	1	1	0	1	1	1	0
R0	1	1	1	0	1	1	0
M2: ↑ {W1	1	0	1	1	1	0	1
R1	1	1	0	1	1	1	1
R1	1	1	1	0	1	1	1
M3: ↓{W1	1	0	1	1	0	0	1
R1	1	1	0	1	0	1	1
R1	1	1	1	0	0	1	1
M4: ↓{ W0	1	0	1	1	0	0	0
R0	1	1	0	1	0	1	0
R0	1	1	1	0	0	1	0
M5: ↕ R0	1	1	1	1	0	1	0
	0	x	x	x	x	x	x

3)Third march element M2 is a multi-operation element, which consists of three operations: i) W1, ii) R1, iii) R1. MUT is addressed in increasing order as

each of these three operations is performed on each memory location before moving on to the next memory location.
4)Fourth march element M3 is a multi-operation element, which consists of Three operations: i) W1, ii) R1, iii) R1. MUT is addressed in decreasing order as each of these three operations is performed on each memory location before moving on to the next memory location.
5) Fifth march element M4 is a multi-operation element, which consists of three operations: i) W0, ii) R0, iii) R0. MUT is addressed in increasing order as each of these three operations is performed on each memory location before moving on to the next memory location.
6)The Sixth march element M5 is a single operation element, which reads zero from memory cells in any order.

IV. FAULTS

Let #O be defined as the number of different operations performed sequentially in a S (Sentizied). Depending on #O, FPs can be divided into static and dynamic faults:

1)Static Faults :

These are FPs which sensitize a fault by performing at the most one operation; that is $\#O \leq 1$. The Static Faults described in this paper are as follows:

a) Deceptive Read Destructive Fault [DRDF]:

A Cell is said to have a DRDF if the read operation performed on the cell returns the expected value while changing the contents of the cell to the wrong value. To detect Deceptive Read Disturb Fault each cell should be read twice successively. The first read sensitize the fault and the second detects it.

R0R0
R1R1

b) Write Disturb Fault:
A Cell is said to have a WDF if a non transition write operation causes a transition in the cell. To detect Write Disturb Fault each cell should be read after a non-transition write.

0W0R0
1W1R1.

2)Dynamic Faults:

These are FPs that perform more than one operation sequentially in order to sensitize a fault; that is #O >1. Depending on #O, a further classification can be made between 2-operation dynamic FPs whereby #O =2, 3-operation dynamic FPs whereby #O =3, etc.
Different types of Single cell Dynamic faults are described below

a)Dynamic Read Destructive Fault (dRDF):
A write followed immediately by a read operation performed on a cell changes the data in the cell, and returns an incorrect value on the output.

<0*w*0*r*0/ 1 /1>

b)Dynamic Deceptive Read Destructive Fault (dDRDF):
A write followed immediately by a read operation performed on a cell changes the data in the cell, and returns a correct value on the output. Here, the write can be a transition write as well as a non-transition write Operation.

<1w1r1/0/1>.

c) Dynamic Incorrect Read Fault (dIRF):

A read operation performed immediately after a write operation on a cell returns an Incorrect value on the output, while the cell remains in its correct state.

<0w0r0/0/1

V. MBISR Operation

The address comparison is done in the redundancy logic, The address is compared to the addresses that are stored in the redundancy word lines, An overflow bit identifies that there are more failing/fault addresses than possible repair cells, The programming of the faulty addresses is done during the memory BIST setup.

The BISR mechanism 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 repair rate and area cost of the Built In Redundancy Analysis (BIRA) is mainly depends on the redundancy organization. The redundancy organization memory is divided into various segments. In which spare row and columns are used differently. Spare rows are used to replace entire row in the memory and the columns are divided in several spare column, groups. Here the access time and area cost is induced due to additional multiplexers. However different redundancy organization will lead to different area cost and repair rate. Since most of the memory faults are single cells, spare words are very efficient in reducing area, more efficient and cost saving.

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.

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, here as 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.

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.

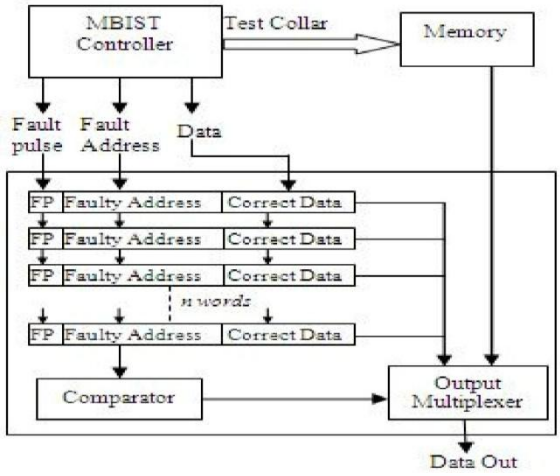

Fig.2 Repair module

The above Fig.2 shows the repair module including the redundancy array and output multiplexer and its interfacing with the existing BIST module.

VI. SIMULATION RESULTS

Verilog HDL Design of Testable SRAM is done with Xilinx ISE Simulator. The design is simulated with same tool ISE Simulator. Simulation Results are shown below. Fig.3 shows the Block diagram of Top Module, and the Simulated Wave form of a Fault free SRAM is shown in Fig.4 .The top module consists of MBIST, Memory, Fault Diagnosis module and redundancy repair array. The waveform resembles the first operation (M0) i.e. write 0 operation which means no data will be read out from memory and WrEna will be '1'.Similarly for remaining operations also we can get simulated waveform. The Faults in SRAM are observed by Introducing Faults. The Write Disturb Fault(WDF),Dynamic Incorrect Read Fault(dIRF) are observed on Fig. 5 at 215ns and 270 ns. The Dynamic Deceptive Read Destructive Fault (dDRDF) is observed in Fig.6 at 1175 ns. The Deceptive Read Destructive Fault DRDF) and Dynamic Read Destructive Fault (dRDF)is shown in Fig.8 at 1650ns and 1685 ns.

Fig.3 Block diagram of Top Module.

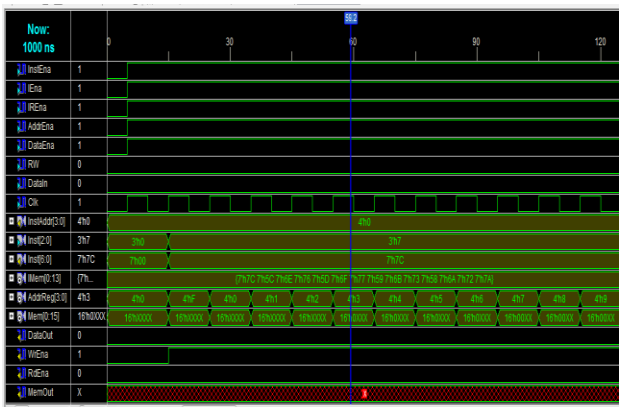

Fig.4:Simulated waveform of Fault-free SRAM.

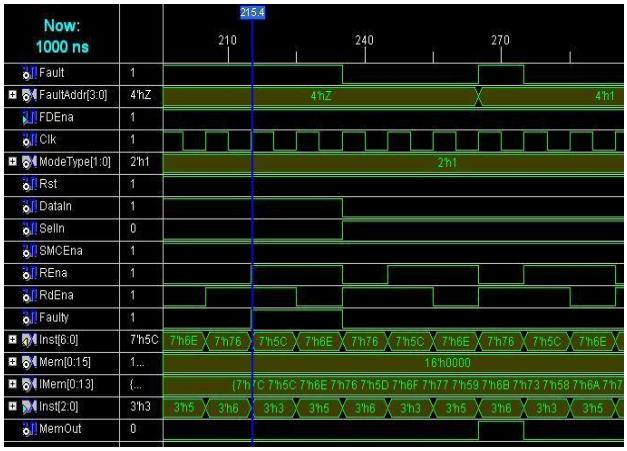

Fig.5: Simulated waveform of Faulty SRAM [WDF, dIRF].

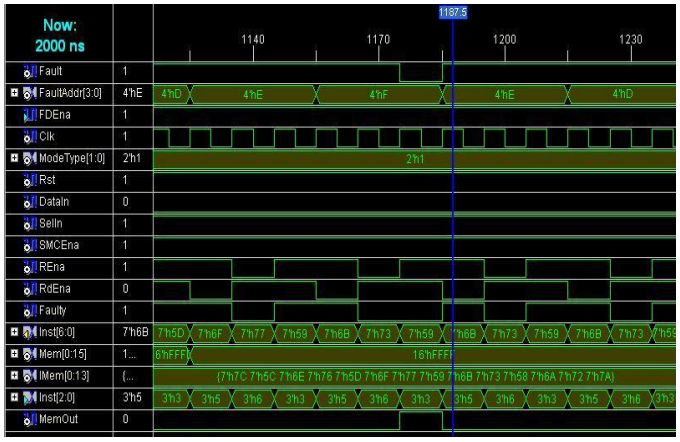

Fig.6: Simulated waveform of Faulty SRAM [dDRDF].

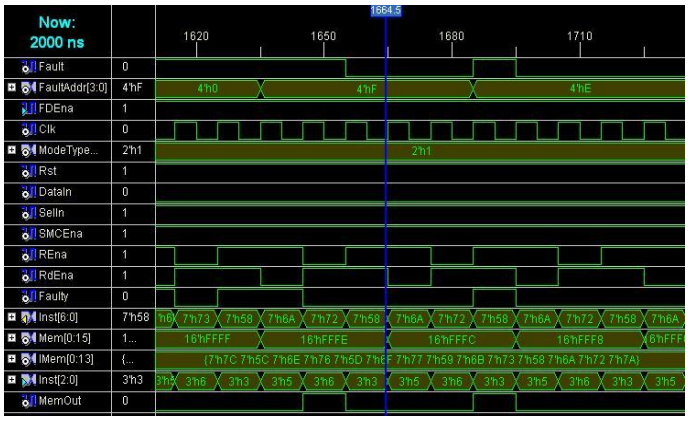

Fig.7: Simulated waveform of Faulty SRAM [DRDF, dRDF].

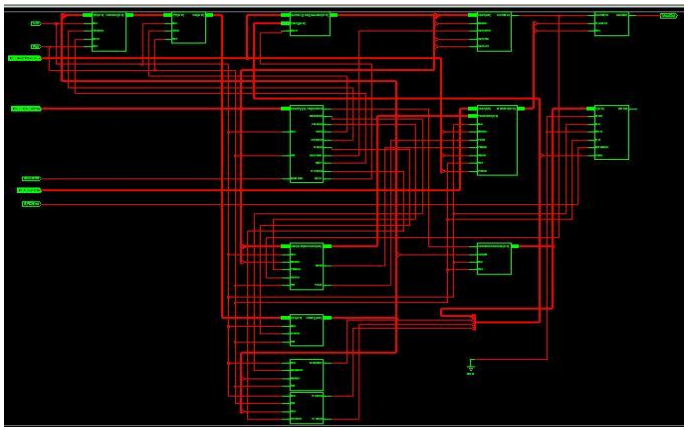

Fig 8 : RTL Schematic for Top Module.

VII. SYNTHESIS REPORT

---- Source Parameters

Input File Name : "TopModule.prj"

Input Format : mixed

Ignore Synthesis Constraint File : NO

---- Target Parameters

Output File Name : "TopModule"

Output Format : NGC

Target Device : xc3s100e-5-vq100

Device utilization summary:

Selected Device: 3s100evq100-5

Number of Slices :	212	out of 960	22%
Number of Slice Flip Flops:	151	out of 1920	7%
Number of 4 input LUTs:	397	out of 1920	20%
Number of IOs:	14		
Number of bonded IOBs:	13	out of 66	19%
Number of GCLKs:	1	out of 24	4%

Timing Summary:

Speed Grade: -5

Minimum period: 7.229ns (Maximum Frequency: 138.328MHz)

Minimum input arrival time before clock: 10.663ns

Maximum output required time after clock: 5.333ns

VIII. CONCLUSION

The Proposed algorithm has been generated to detect and diagnose the Static as well as Dynamic faults in SRAM have been successfully implemented, Xilinx ISE 9.1i is used to verify the functionality and timing Constraints of Verilog Coded BIST Module, Repair redundancy and their interface, and all modules are synthesized using Xilinx ISE 9.1i and synthesis report is generated. The proposed algorithm has less no of instructions then the previous one i.e. March SS this leads to Micro Instructions Optimization, which in turn Reduces the time required to detect the faults . The faults which are not covered by previous algorithm is covered in this paper.

REFERENCES

[1] Dr. R.K. Sharma and Aditi Sood "Modeling and Simulation of Multi-operation Microcode-based Built-In Self Test for Memory Fault Detection and Repair",2010 IEEE Annual Symposium on VlSI,DOI 10.1109/ISVLSI.2010.88.

[2] 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.

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

[4] 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.

[5] Dongkyu Youn,Taehyung Kim, Sungju Park,"A microcode-based memory BIST Implementing modified march algorithm ",Asain Test Symposium,2001. Proceedings. 10th, 2001,pp.391-395.

[6] Schober,V.;paul,s.;picot,o.;"Memory built-in-self-repair using redundant Words," International TestConference,2001.(ITC 2001).proceedings,2001.pp.995-1001.

[7] R. Dekker, *et al.*, \A Realistic Fault Model and Test Algo rithms for Static Random Access Memories", *IEEE Trans on Computers*, C9(6), pp. 567-572, 1990.

[8] R.D. Adams and E.S. Cooley, \Analysis of a Deceptive Read Destructive Memory Fault Model and Recommended Test ing", *In Proc. IEEE North Atlantic Test Workshop*, 1996.

[9] M.S Abadir and J.K. Reghbati, "Functional Testing of Semiconductor Random Access Memories ,"ACM Computer Surveys, vol 15,no.3 pp, 175-198,1983.

[10] Ch. E. Stroud, "A Designer's Guide to Built-In Self Test", Kluwer Academic Pubs., ISBN 1-4020-7050-0, 2002.

[11] 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.

[12] 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.