ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 10, October 2014

Microcode March SS Algorithm Implementation in the MBIST with Redundancy Repair Module

J. Praneet Raj

Assistant Professor, Department of Information Technology, Vasavi College of Engineering

Abstract: There is a huge exponential growth in the area occupied by the embedded memories on the System-on-Chip (SoC) because of the data that the users prefer to store in the devices. The expectation of the user is that it has to be very precise and accurate at the same time. Hence there is a need for testing these modules once in a while for which Built in Self Test was introduced in recent years. Testing is only an initial part after which comes the repairing where the faults detected during testing must be answered. In this paper March SS algorithm is implemented to test an embedded memory in such a way that the steps in the algorithm is converted into microcode that is the most optimized way of implementing a algorithm that is used for testing in BIST module. Along with BIST an enhancement where we can repair the faulty address locations is included which does the repairing by the means of a Redundancy module. Due to this we can have the best implementation where we can have a testing as well as repairing the faulty address location in a single SoC. The proposed architecture was implemented using Verilog HDL, simulated using Xilinx simulation tool and synthesis by Xilinx Synthesis tool.

Keywords**:** Built in Self Test (BIST), Redundancy module, March SS algorithm.

I. INTRODUCTION

Now a days, the area occupied by embedded memories in System-on-Chip (SoC) is over 90%, and expected to rise up to 94% by 2014 [2]. As the size of the chip reduces the scope of making an error is more. The faults in these embedded memories are increasing day by day. New fault models have to be developed to eliminate these faults. Higher the fault detection capability, of the March algorithm higher the possibility of repair.

II. MARCH ALGORITHM

The algorithms implemented in Memory Built in Self Test (BIST) vary in the number of elements and also the number of operations in those elements, commonly called as March Algorithms. There are only two operations in these algorithms which are writing and reading. The write operations perform a data write into the memory where as a read operation perform the read operation on the memory. The way the memory module is accessed is also given from the algorithm, ↑ specifies up (accessing the memory locations from lower address to the higher address), ↓ specifies down (accessing the memory locations from highest address to the lowest address), whereas ↕ specifies bidirectional which means its user defined the memory location can be accessed in any direction depending upon the user. Many different algorithms are present but choosing a highly efficient algorithm plays a very critical action. The time required to test a particular memory module depends upon the total number of March operations. The designer has to choose in such a way that time required to test must be less whereas the number of faults detected must be more. Here we have categorized few algorithms and choose the best suitable algorithm.Test length gives the total number of operations in the algorithm. Fault coverage shows the different faults that can be detected once we implement the specific algorithm. From the above table we can see that March SS algorithm has a total of 22 operations and is also able to detect more number of faults which can be chosen as a suitable algorithm when we have a SoC which is used for a very complex application.

TABLE I
COMPARISON OF DIFFERENT MARCH ALGORITHMS

Algorithm	**Test Length**	**Fault Coverage**
MATS +	5n	SF, RDF, IRF
MARCH C-	10n	SF, TF, RDF, IRF, CFst, CFds, CFtr, CFrd, CFir
MARCH B	17n	SF, TF, RDF, IRF, CFds
MARCH SS	22n	SF, TF, WDF, RDF,DRDF, IRF, CFst, CFds, CFtr, CFwd, CFrd, CFdrd, CFir

III.PRIOR WORK

Many algorithms are implemented in this field of study where we can detect the faults in an embedded memory. In the prior architecture of microcode the algorithm that was implemented had 46 operations due to which the time taken to complete the testing operation. That architecture was able to perform only the testing hence it was able to detect only the faults in the memory under test. In this paper the architecture is modified in such a way that after testing the memory under test we are able to perform a

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 10, October 2014

repairing operation with the help of a redundancy module. Hence this architecture has a added advantage of testing as well as repairing.

IV. PROPOSED ARCHITECTURE

The figure1 below shows the proposed architecture. The main modules of this architecture are it consists of a Clock generator, Instruction pointer, Microcode Instruction Storage, Instruction register, Memory under test, comparator, multiplexer modules and Redundancy.

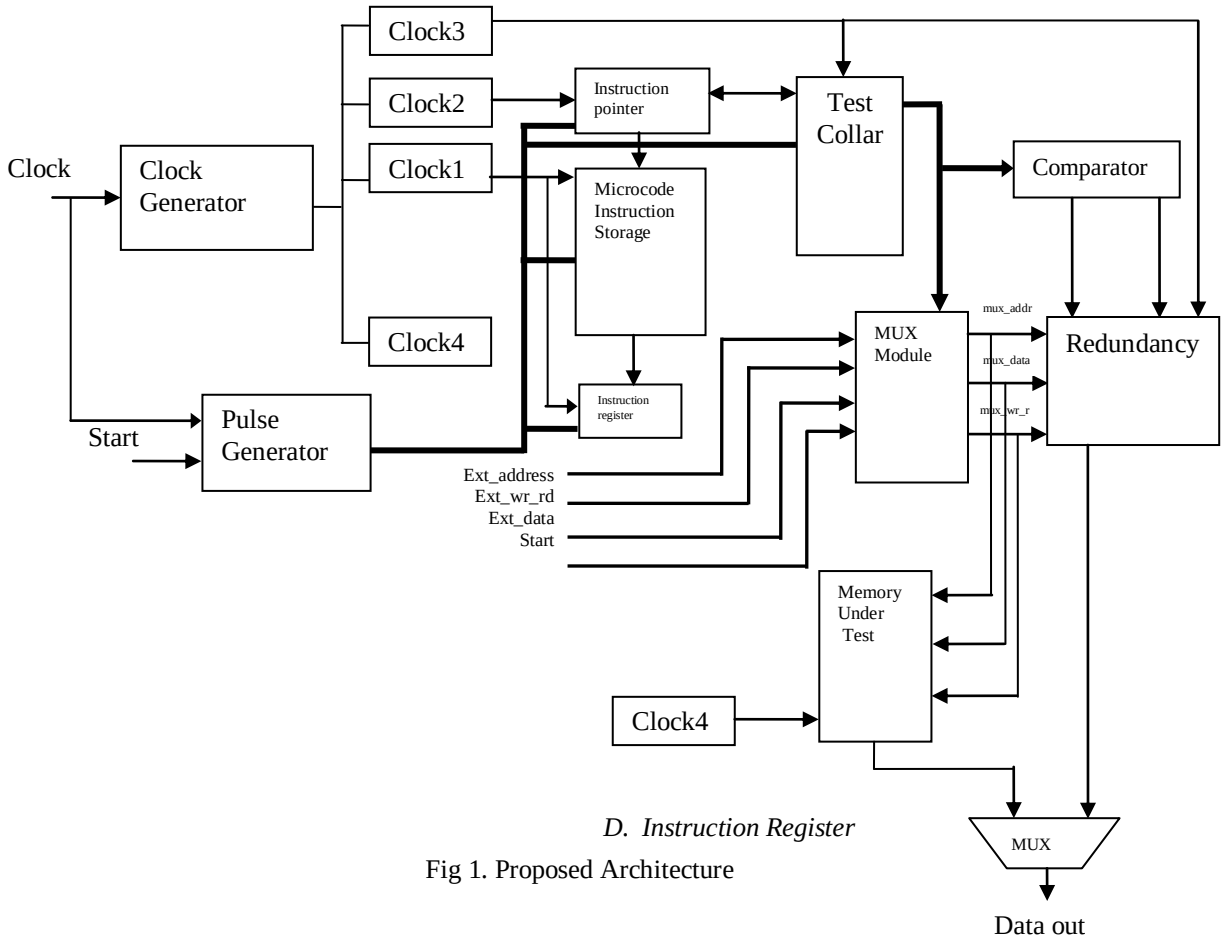

Fig 1. Proposed Architecture

A. Clock Generator

The clock generator is implemented to generate different clocks for providing them to internal modules in order to synchronize. The clock generator generates four different clocks namely clock1, clock2, clock3, clock4.

B. Pulse Generator

The pulse generator is a simple module which implements latch architecture. This module generates the start pulse depending on a input signal. Whenever a user wants to start the testing of a memory module the input is made high this is provided to different modules in the architecture which starts implementing the March SS algorithm.

C. Instruction Pointer

It points to the next Microword, which 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.

D. Instruction Register

This 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.

E. Multiplexer Module

The Multiplexer Module in the proposed architecture has 7 different inputs. Three of these inputs are from the Test Collar outputs and the other three are external inputs. The external inputs are the Address, Data and write enable. The inputs to the RAM are Address, Data and control logic. The selection of test collar inputs or the external inputs is dependent on the start signal.

F. Test Collar

The Test Collar internally consists of 3 modules for these test pattern generations such as Address generator, Data

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 10, October 2014

generator, Write/Read enable generators. The outputs from these generators are given to the multiplexer module.

G. Comparator

The comparator module works during the test mode to give the fault signals 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.

H. 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 [1]. 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 Microword.

#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 2. Microcode Format

A 7-bit microcode instruction is formed. Bit 1 indicates a valid microcode instruction, Bit 2, Bit 3 and Bit 4 stand for first operation, in-between operation and last operation of a multi-operation March element. Bit 5 specifies the way in which memory is being accessed (Increment / Decrement). Bit 6 defines whether a read operation or a write operation is being performed on the memory. Bit 7 specifies the bytes of 1s that are to be generated while writing or the expected result while reading.

I. Redundancy

The redundancy module consists of three different fields such as address field, data field, flag field. The faulty addresses detected during the read operations of the March elements will be stored in these address fields.

During the normal operation the addresses that are being provided by the user might be faulty addresses, the data that are to be stored in these faulty address will be stored in the data fields in this redundancy instead of storing them in the faulty locations and after storing the data a flag will be set in the flag fields.

During the read operation under normal mode when data are to be read out data that is stored in these redundancy data fields will be directed through an output multiplexer for which the select line is given by the flag. If no faults arise then this redundancy module is inactive. To enable this module fault signal from the comparator is given to this.

	#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

Fig 3. Microcode March SS algorithm

V. SIMULATION RESULTS

In the figure 4 we see that Tc_RunComplete is made high when the complete test collar is implemented. Whenever Start signal goes down that mean that external operation is being started and the address, data and write/read must be provided by the user. During the testing operation the faulty addresses that are detected are 7,6,4,0 as we can see near the addr_field signal in the fig. In the normal mode user starts giving some data to be stored in the memory data 22 must be stored in address 2, data 44 must be stored in address 4, data 66 must be stored in address 6, data 77 must be stored in address 7 and data 88 must be stored in address 0. The control signal is also provided as write on the Ext_wr_rd signal.Since the locations 7,6,4,0 are faulty the data 77,66,44,88 are stored in the data field in the redundancy instead of MUT. When this Ext_wr_rd signal is low it means that read operation has to be done on the MUTThis is finally given on data_out. Hence we see that the repair is implemented accordingly with the help of redundancy module implementation.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 10, October 2014

Four faults are introduced in the memory locations 0, 4, 6, 7. This is detected during the read operations in the March SS algorithm. These faulty addresses are stored in the address field in the Redundancy module as shown in the figure5. During the external write operations the data that are to be stored in these faulty locations will be stored in the corresponding data fields as shown in the fig and respective flag fields are set.

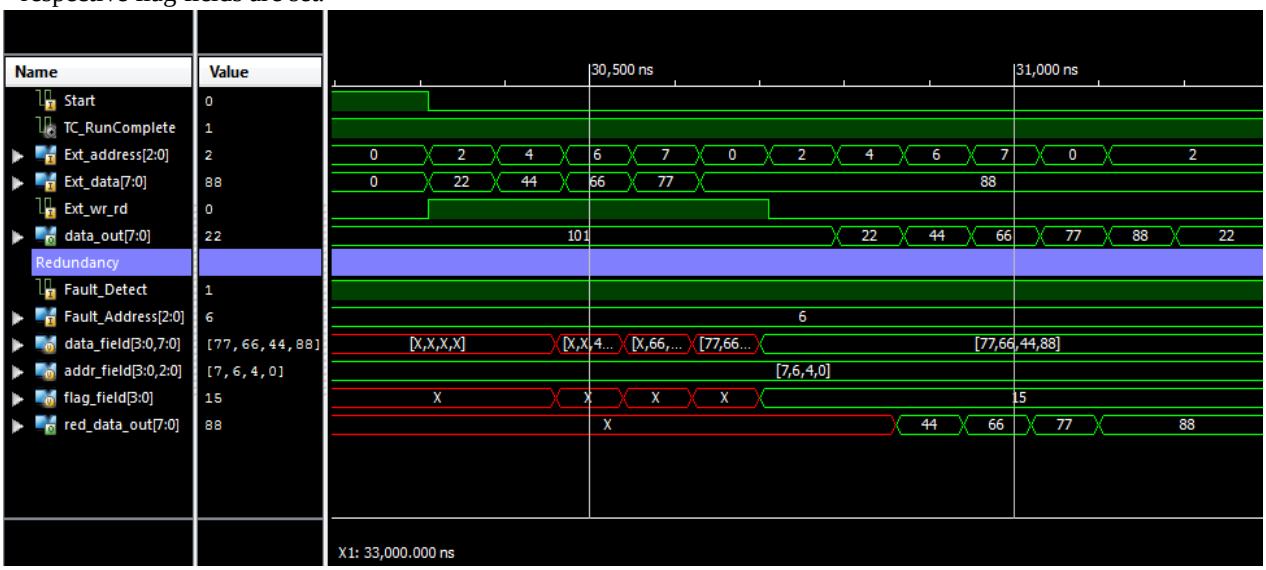

Fig 4. Simulation Result showing various signals in the architecture

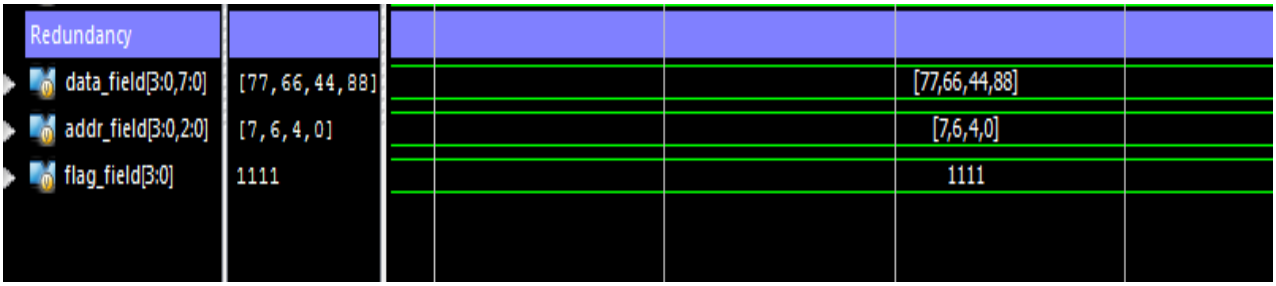

Fig 5. Simulation Result of Redundancy Module

VI. Synthesis Report

Figure 6 Shows the Synthesis Report where the total number of Flip-flops and Look-up-tables (LUTs) are utilized in the Spartan 3E FPGA.

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	164	4656	3%
Number of Slice Flip Flops	188	9312	2%
Number of 4 input LUTs	283	9312	3%
Number of bonded IOBs	27	232	11%
Number of GCLKs	3	24	12%

Fig 6. Synthesis Report showing the Device Utilization Summary

VII. Conclusion

The simulation results have shown that the Microcode March SS algorithm is implemented successfully and is integrated with a redundancy module such that faulty addresses are stored and repaired on the chip itself. The redundancy module uses spare locations where we can store data also. This can be easily implemented in the System-on-Chips (SoCs).

References

[1] Prof.Vinod Kapse, Mohammed Arif, "Optimization of Microcode Built-In Self Test By Enhanced Faults Coverage for Embedded Memory", 2012 IEEE Students Conference on Electrical, Electronics and Computer Science.

[2] Semiconductor Industry Association, "International technology roadmap for semiconductors (ITRS), 2003 edition," Hsinchu, Taiwan, Dec.2003.

[3]. Said Hamdioui, Ad J. van de Goor, Mike Rodgers, "March SS: A Test for All Static Simple RAM Faults", Proceedings of the 2002 IEEE International Workshop on Memory Technology, Design and Testing (MTDT 2002)

BIOGRAPHY

J. Praneet Raj, M.E. (Embedded Systems and VLSI Design) is presently working as Assistant Professor in the Department of Information Technology in Vasavi College of Engineering. He has presented 3 papers in International conferences 1 paper in a National Conference. His field of interest is VLSI Design.