Modeling and Simulation of Built-in Self Repair Technique for Memories

J Arun Kumar [1] , J.Narasimha Rao [2]

[1]*Department of ECE, Narasaraopet Engineering College, Narasaraopet, Ap, India.*
[2]*Associative proffesor Department of ECE, Narasaraopet Engineering College, Narasaraopet, Ap, India.*

Abstract: Built-in self repair (BISR) techniques have been widely used to enhance the yield of embedded memories. As embedded memory area on-chip is increasing and memory density is growing, problem of faults is growing. So that new test algorithms are developed for detecting these new faults. These new march algorithms have much more number of operations than the March algorithms existing earlier. March ss algorithm is used to implement architecture which detects faults, known as built-in self test (BIST) module. These detected faults can be repaired by using built-in redundancy analysis (BIRA) module presented in the built-in self repair architecture.

Keywords- Built-in self test (BIST), Built-in redundancy analysis (BIRA), Built-in self repair (BISR).

I. INTRODUCTION

In modern system-on-chip (SOC) designs, embedded memories occupy a significant portion of the chip area [1]. According to the 2001 ITRS, today's systems on chips (SoCs) are moving from logic dominant chip to memory dominant chips in order to deal with todays and future application requirements. The dominating logic (about 64% in 1999) is changing to dominating memory (approaching 90% by 2011) [2],[3] as shown in Figure.1.

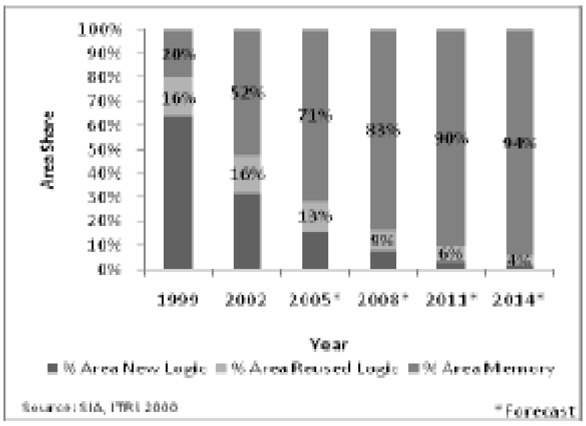


Figure1. The future of embedded memory

These shrinking technologies give rise to new defects so new fault models have to be defined to detect and eliminate these new defects. These new fault models are used to develop new high coverage test and diagnostic algorithms. The greater the fault detection and localization coverage, the higher the repair efficiency; hence higher the obtained yield. Memory repair is necessary, since just detecting the faults is no longer sufficient for SoCs, hence both diagnosis and repair algorithms are required. March SS [5] and March RAW [4] are examples of two such newly developed test algorithms that deal with detecting some recently developed static and dynamic fault models.

II. PROPOSED BISR SCHEME

The BISR circuit mainly consists of a built-in self-test (BIST) module, BIRA module, and a reconfiguration mechanism. The BIST performs the applied test algorithm and sends the fault information of the detected faults to the BIRA module for analysis. The reconfiguration mechanism can swap the defective elements with the redundancies according to the redundancy allocation results of the BIRA.

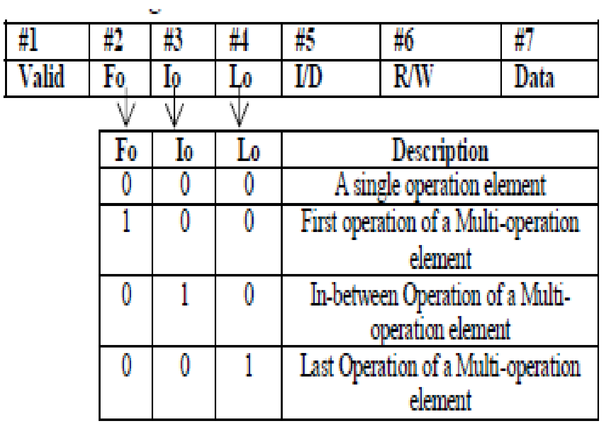
A. PROPOSED BIST ALGORITHM

The DPM screening of a large number of tests applied to a large number of memory chips showed that many well known fault models, developed before late 1990s failed to explain the occurrence of complex faults. This implied that new memory technologies involving high density of shrinking devices lead to newer faults. This stimulated the introduction of new fault models, based on defect injection and SPICE simulation. Some such newly defined fault models [6] are Write Disturb Fault (WDF), Transition Coupling Fault (Cft), Deceptive Read Disturb Coupling Fault (Cfdrd). Another class of faults called Dynamic faults which require more than one operation to be performed sequentially in time in order to be sensitized have also been defined. [4], [7]. These new faults cannot be easily detected by established tests like March C, rendering it insufficient/ inadequate for today's and the future high speed memories. More appropriate test algorithms like March SS and March RAW have been developed to deal with these new fault models. While March SS covers some of the new fault models like Deceptive Read Destructive fault (DRDF), Write disturb fault (WDF), etc., March RAW covers some of the Dynamic faults.

The test elements M1 through M4 of March SS algorithm have five test operations per element. This is in contrast with some of the algorithms developed earlier like March B, MATS+, March C which only had up to two operations per March element [9].

MICRO CODE SPECIFICATIONS:

The microcode is a binary code that consists of a fixed number of bits, each bit specifying a particular data or operation value [8]. The microcode instruction developed in this work is coded to denote one operation in a single micro word. Thus a five operation March element is made up by five micro-code words. The format of 7-bit microcode BIST Instruction word is as shown in Fig. 2. Its various fields are explained as follows: Bit #1 (=1) indicates a valid microcode instruction, otherwise, it indicates the end of test for BIST Controller. Bits #2, #3 and #4 are used to specify first operation, in-between operation and last operation of a multi-operation March element, interpreted as shown in Figure 2.



#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

Fig2: Format of Microcode Instruction

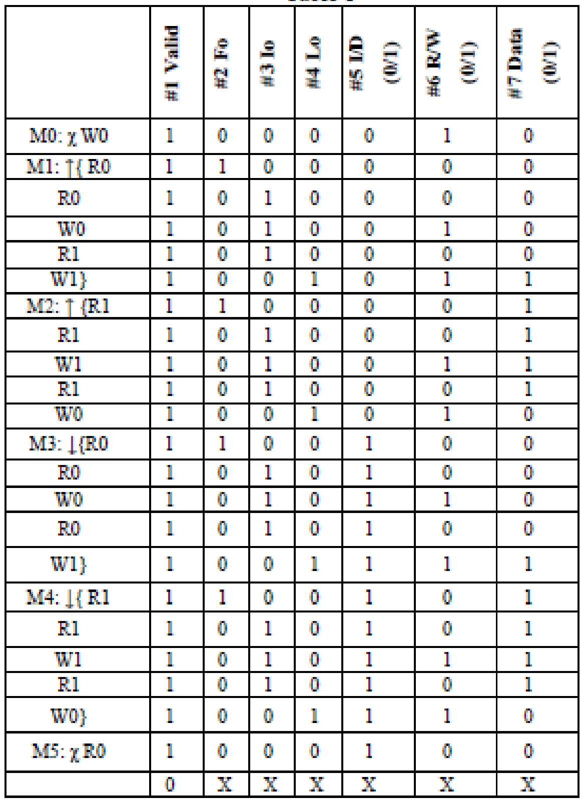
Bit #5 (=1) notifies that the memory under test (MUT) is to be addressed in decreasing order; else it is accessed in increasing order. Bit #6 (=1) indicates that the test pattern data is to be written into the MUT; else, it is retrieved from the memory under test. Bit #7 (=1) signifies that a byte of 1s is to be generated (written to MUT or expected to be read out from the MUT); else byte containing all zeroes are generated.

The instruction word is designed such that it can accommodate any existing or future March algorithm. The contents of Instruction storage unit for March SS algorithm are shown in Table 1.

The first March element M0 is a single operation element, which writes zero to all memory cells in any order, whereas the second March element M1 is a multi operation element, which consists of five operations: i) R0, ii) R0, iii) W0, iv) R1 and v) W1. MUT is addressed in increasing order as each of these five operations is performed on each memory location before moving on to the next location

.

Table 1



	#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

B. BIRA IMPLEMENTATION

Fig. 3 shows the block diagram of the proposed BIRA, which consists of a FSM, a local bitmap, and repair signature registers. The FSM realizes the redundancy allocation procedure of the proposed redundancy analysis algorithm [1]. The local bitmap stores the faulty information of detected faults and checks if the detected fault has been repaired or stored. The repair signature registers store the repaired addresses. Subsequently, we briefly explain the operation of the proposed BIRA. If the BIST detects a fault, it is paused and issues a fail signal to activate the BIRA circuitry. Then the BIRA catches the fault information of the fault through the faulty address and faulty syndrome ports. Subsequently, the BIRA performs the redundancy analysis procedure. Once the BIRA completes the redundancy analysis procedure, it informs the BIST to resume the test process through the continue signal. During the BIRA process, the repaired addresses are stored in the repair signature registers. After the test is finished and the Local bitmap is clear, the BIRA asserts the ra_finish signal. Then the repair signature in the registers can be exported to the external equipment or the fuse box through the rsr_out Port by asserting the shift_en signal. If the memory cannot be repaired, the unrepairable signal is asserted. Finally, if a memory is unrepairable, the user can program the threshold value through the program signal and repeat the repair process.

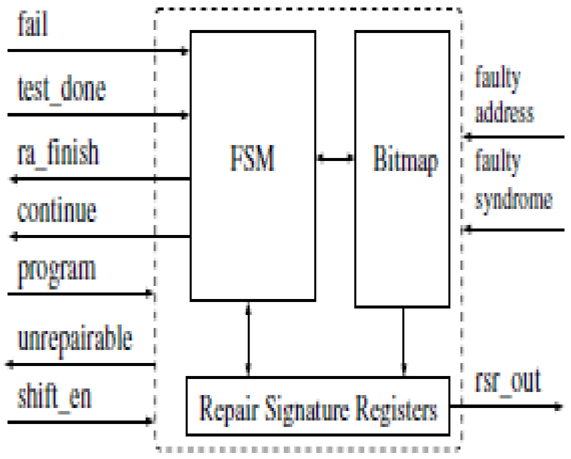


Fig3: Block diagram of proposed BIRA

III. BISR ARCHITECTURE

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 March SS 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 block diagram of the BISR controller architecture shown in Figure 4. The BIST Control Circuitry consists of Instruction Pointer, Microcode Instruction storage unit, Instruction Register. The Test Collar circuitry consists of Address Generator, RW Control and Data Control.

Instruction Pointer points to the next micro word that 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.

Instruction Register holds the micro word (containing the test operation to be applied) pointed at by the Instruction Pointer. The various relevant bits of micro word are sent to other blocks from IR.

Address Generator points to the next memory address in MUT, according to the test pattern sequence. It can address the memory in forwards as well as backwards direction.

RW Control generates read or write control signal for MUT, depending on relevant micro word bits.

Data Control generates data to be written to or expected to be read out from the memory location being pointed at by the Address Generator. The Ad dress generator, RW Control and Data Control together constitute the Memory *Test Collar*

Input Multiplexer directs the input to memory by switching between 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 from the Test Collar. In case of Normal mode the memory responds to the external address, data and read/write signals.

Fault Diagnosis module works during the test mode to give the fault waveform which consists of positive pulses 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.

Redundancy array logic consists of signature registers and its corresponding spare registers. Signature registers stores the faulty addresses during the test mode. During the normal mode each incoming address is compared with the address fields stored in the signature registers.
If there is match found, the data corresponds to the faulty address is stored in the respective spare register. The correct data in the spare register can be read by using read operation in the normal mode.

Output multiplexer of redundancy array logic give the output when performing read operation in normal mode.

State machine controller controls every operation in the BISR architecture. Based on the operation it activates the required blocks and deactivate the remaining blocks.

.

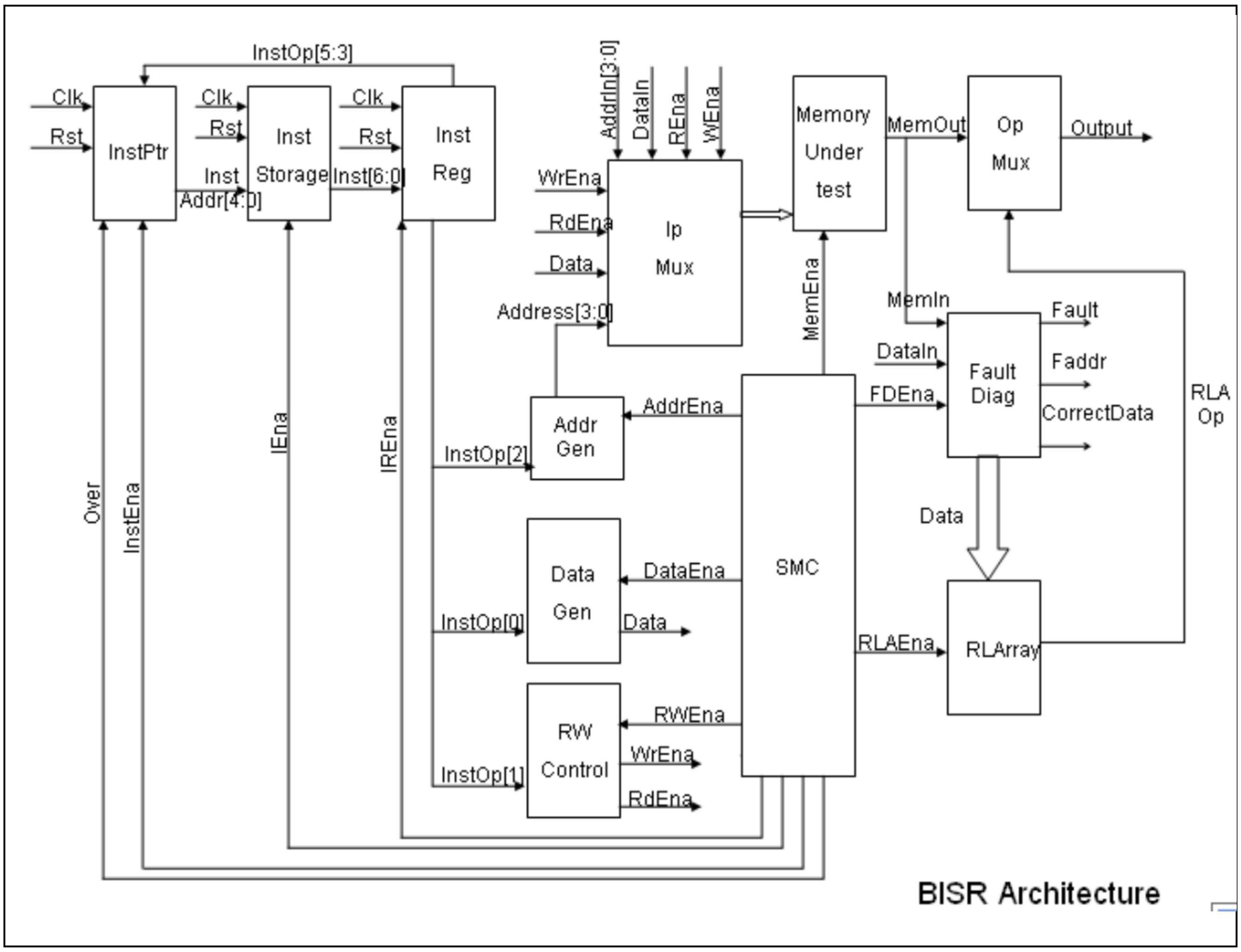


Fig4: BISR Architecture

IV.RESULTS

ModelSim simulator has been used to verify the functionality and timing constraints of Verilog coded BISR module Repair redundancy array, and their interface.
The full architecture containing all these modules has been successfully synthesized using Xilinx ISE 9.1i. Design Synthesis Report shows that a total of 840 slices, 936 slice flip-flops, 1114 4 input LUTs and 39 bonded IOBs have been used in the synthesis.
The simulation waveforms of faulty memory shown in the figure5. Here top module results shown for mode one, that is in this mode faults will be identified using marsh ss algorithm. Based on this each and every memory location checked for faulty locations by performing consecutive read and write operations. That is each time algorithm reads or writes the data into the memory. In the above shown figure 5 DATAIN signal specifies the input data given to the memory, MEMIN is output signal from memory under test and it is input to the fault diagnosis block. Both DATA IN and MEMIN signals are compared, after read operation is performed from memory under test. If any mismatch is Found in the data of both signals than fault signal will be raised after next clock pulse of read signal. That particular address of memory under test will be considered as faulty location. These faulty locations of memory under test are identified in the test mode (BIST module). These faulty locations are stored in the signature registers in the test mode.

The redundancy analysis results are shown in the figure6. In this mode each signature register consist of its corresponding spare registers, which holds correct data of faulty location. When ever we are trying to write the data into the memory, externally giving address, compare with the address of signature registers, if match found that particular location is faulty location, so instead of storing data in faulty location we store the data in spare registers corresponding to that signature register. So when we are performing read operation , it compare the external address with the faulty address in the signature registers, if match found it reads the data from spare register corresponds to that particular signature register, if not it reads data directly from memory locations shown in the figure 6. This module is redundancy analysis module (BIRA).

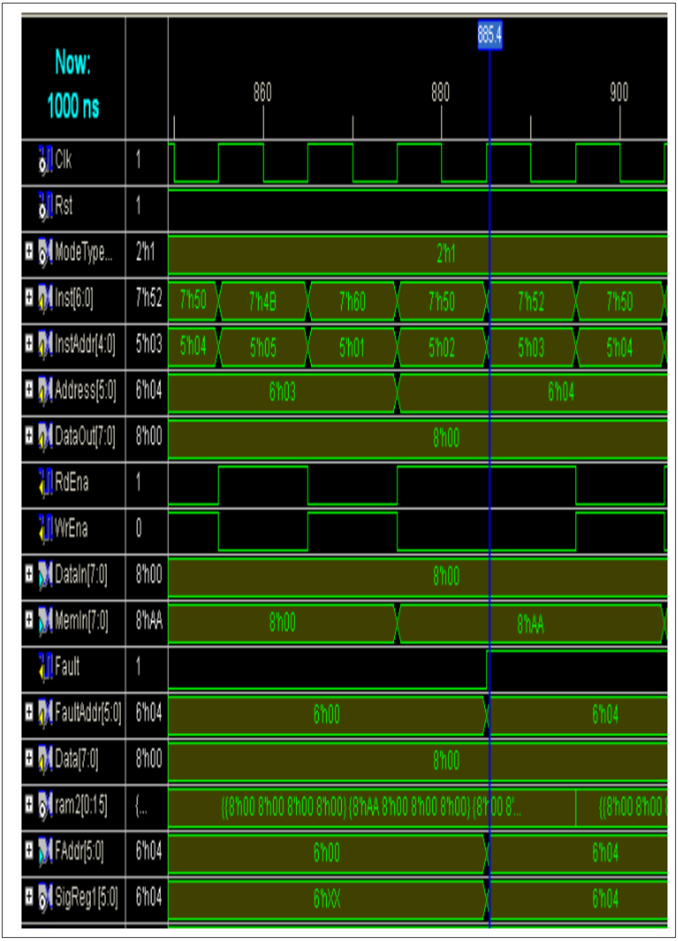


Fig5: simulated waveform of faulty memory

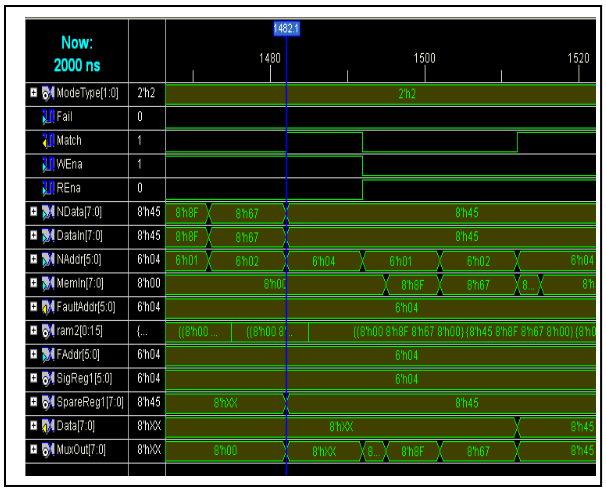


Fig6: simulated wave form of BIRA

V.CONCLUSION

The simulation results have shown that the Built-in Self Repair (BISR) architecture is successfully able to implement new test algorithms. Implementation of a single test operation in one micro word ensures that any future test algorithms with any number of test operations per test element are successfully implemented using the current BISR architecture. Moreover, it provides a flexible approach as any new march algorithm, other than March SS can also be implemented using the same BISR hardware by changing the instructions in the microcode storage unit, without the need to redesign the entire circuitry. A detailed power, time and area overhead analysis of this architecture is underway and efforts are being made to develop a power-optimized BISR architecture for embedded memories [10].

REFERENCES

[1] Yi-Ju Chang, Yu-Jen Huang, and Jin-fu Li ”a built-in redundancy analysis scheme for rams with 3d redundancy” IEEE,2011.
[2] R.k Sharma, aditi sood “Modelling and Simulation of multi-operation microcode-based Built-in self test for memory fault detection and repair”. *In Proc. of IEEEVLSI annual Symposium*, pp. 381-386, 2010.
[3] International SEMATECH, “International Technology Roadmap for Semiconductors (ITRS): Edition 2001”
[4] 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.
[5] 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.

[6] 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.
[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] R. Dean Adams, “High Performance Memory Testing: Design Principles, Fault Modeling and Self-Test”, Springer US, 2003.
[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*
[10] C.-T. Huang, C.-F. Wu, J.-F. Li, and C.-W. Wu, “Built-in redundancy analysis for memory yield improvement,” IEEE Trans. on Reliability, vol. 52, no. 4, pp. 386-399, Dec. 2003.