Implementation of Memory Test Controller and design of Control Logic for repair module

Deepa.V.H, (1DA04LVS03), MTech, Dr. Ambedkar Institute of Technology

ABSTRACT

Testing semiconductor memories is increasingly important today because of the high density of current memory chips. This paper presents an overview of testing and repairing of semiconductor random access memories (RAMs). An important aspect of this test procedure is the detection of permanent faults that cause the memory to function incorrectly and the control logic of the repair module that can be used. Functional-level fault models are very useful for describing a wide variety of RAM faults. Several fault models are discussed throughout the paper. Test procedures for these fault models are presented which are widely used today for testing chip level, array level and board level functional memory defects. Repairing of the memory includes replacing the rows and columns which have maximum errors with the redundant rows and columns present in the memory.

I. INTRODUCTION

The standard process for repairing the memories involve three phases.Firstly, the memories are tested and faulty cells are located. Secondly, the rows and columns which have the maximum errors and which have to be replaced are selected. Thirdly, the addresses of rows and columns are given to the computer controlled laser to disconnect the defective rows and columns and connect the spare rows and columns. Since third one is just the hardware, this paper concentrates on the first two steps. Through the years different approaches have been investigated and proposed for testing memories. The most traditional approach is to simply apply a sequence of test patterns to the I/O pins and test the functionality of the memory. This paper concentrates on Marching 1/0 test [*Breuer & Friedman, 1976*][5], MATS test [*Nair, Thatte & Abraham, 1979*][8], MATS+ test[*Abadir & Reghbati, 1983*], MATS++ [*Goor, 1991*],MARCH X [*unpublished*], MARCH C [*Marinescu, 1982*][10], MARCH C- [*Goor, 1991*], MARCH A [*Suk & Reddy, 1981*], MARCH Y [*unpublished*],MARCH B [*Suk and Reddy*, [1981][9], The main goal behind these approaches is to reduce the memory testing time which rises exponentially with memory size. Along with these tests a repair module design is also provided for the memory which replaces the rows and columns with maximum errors with redundant rows and columns.

II. MEMORY FAILURE MODES

Classical fault models are not sufficient to represent all important failure modes in a RAM; Functional Fault models should be employed. Memory Fault models can be classified under the categories shown below, brief descriptions of the models are given as follows.

Functional Fault models:

1. *Stuck-at fault (SAF):* cell or line s-a-0 or s-a-1
2. *Stuck-open fault (SOF):* open cell or broken line.
3. *Transition fault (TF):* cell fails to transition from one state to another.

4. *Data retention fault (DRF):* cell fails to retain its logic value after some specified time due to, e.g., leakage, resistor opens, or feedback path opens

5. *Coupling fault (CF):* Coupling Faults are of three types

- Inversion coupling fault (CFin): a transition in one cell (aggressor) inverts the content of another cell (victim).
- Idempotent coupling fault (CFid): a transition in one cell forces a fixed logic value into another cell.
- State coupling fault (CFst): a cell/line is forced to a fixed state only if the coupling cell/line is in a given state (a.k.a. pattern sensitivity fault (PSF)).

6. *Bridging fault (BF):* short between cells (can be AND type or OR type)

7. Neighborhood Pattern Sensitive Fault (NPSF)

8. Active (Dynamic) NPSF

9. Passive NPSF

10. Static NPSF

Address decoder Faults *(AFs)*

1. No cell accessed by certain address.

2. Multiple cells accessed by certain address

3. Certain cell not accessed by any address

4. Certain cell accessed by multiple addresses

For the sake of simplicity, the dynamic faults are not considered.

III. ALGORITHM'S AND ANALYSIS

A MARCH TEST consists of a finite sequence of March elements, while a March *element* is a finite sequence of operations applied to every cell in the memory array before proceeding to the next cell. An o*peration* can consist of writing a 0 into a cell (w0), writing a 1 into a cell (w1), reading an expected 0 from a cell (r0), and reading an expected 1 from a cell (r1).

MARCHING 1/0 Test:

The MARCHING 1/0 is a Test of 14n complexity. It is a complete Test for AF's, SAF's and TF's but has the ability to detect only a part of CF's . The Test sequence is given as follows.

MATS Test:

MATS stands for Modified Algorithmic Test Sequence. MATS is the shortest March test for unlinked SAF's in memory cell array and read/write logic circuitry. The algorithm can detect all Faults for OR type technology since the result of reading multiple cells is considered as an OR function of the contents of those cells. This Algorithm can also be used for AF's of AND type technology using the MATS-AND Test sequence given below . The MATS Algorithm has a complexity of 4n with a better fault coverage compared to equivalent zero-one and checkerboard tests.

MATS+ Test:

The MATS+ test sequence detects all SAF's and AF's, its often used instead of MATS when the technology used under test is unknown. The MATS+ algorithm has a test complexity of 5n.

MATS++ Test:

The MATS++ Test sequence is a complete, irredundant, & optimized Test sequence. It is similar to the MATS+ Test but allows fault coverage for TF's. Recommended test of 6n Test complexity for unlinked SAF's and TF's.

MARCH X :

The MARCH X Test is called so since it has been used without being published [3]. This test detects unlinked SAF's, AF's, TF's and CFin's. The MARCH X test is a test of 6n complexity.

MARCH C :

The MARCH C Test is suited for AF's, SAF's, TF's and all CF's [3]. It is a test of 11n complexity.

MARCH C:

This Test sequence is a modification to March C test implemented in order to remove redundancy present in it. Detects unlinked AF's, SAF's, TF's and all CF's. This test is of complexity 10n.

MARCH A:

The MARCH A Test is the shortest test for AF's, SAF's, linked CFid's, TF's not linked with CFid's, and certain CFin's linked with CFid's [2]. It is a complete and irredundant test of complexity 15n.

MARCH Y :

MARCH Y Test is an extension of March X. This test is of complexity 8n and can detect all faults detectable by March X.

MARCH B:

The MARCH B Test is an extension of MARCH A Test. It is a complete and irredundant test capable of detecting AF's, SAF's, linked CFid's or CFin's. This test is of complexity 17n.

Fault Coverage for March tests:

All the algorithms are implemented in VHDL and select lines are provided externally which is used for the selection of test algorithm.The memory test controller is built with all these test algorithms and the flow charts used for this is given in Fig 1.

Fault	MATS++	MARCHX	MARCHY	MARCH C-
SAF's	100%	100%	100%	100%
TF's	100%	100%	100%	100%
SOF's	100%	0.2%	100%	0.2%
AF's	100%	100%	100%	100%
CFin's	75.0%	100%	100%	100%
CFid's	37.5%	50.0%	50.0%	100%
CFst's	50.0%	62.5%	62.5%	100%

Table 1: Fault coverage in march tests

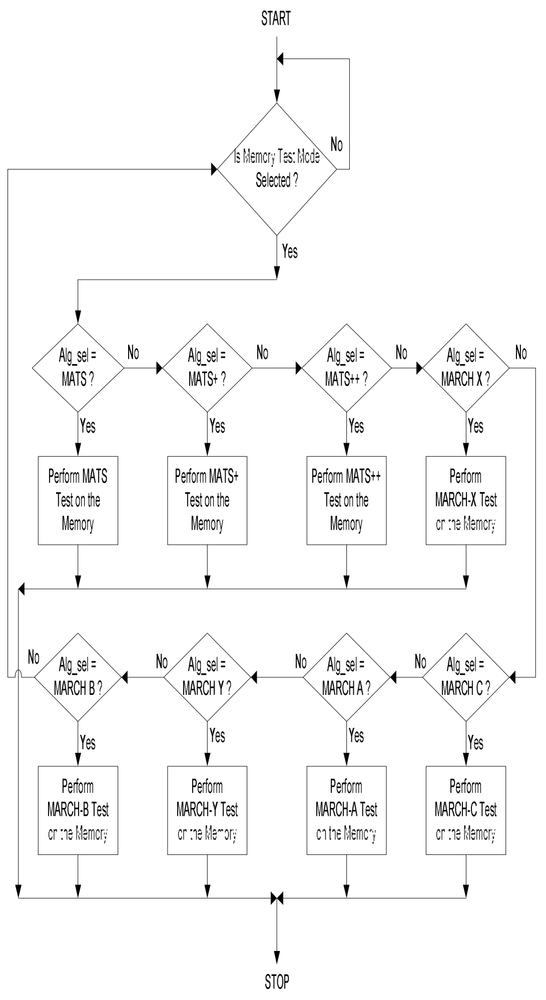
START
Is Memory Test Mode Selected ?
No
Yes
Alg_sel = MATS ?
Alg_sel = MATS+ ?
Alg_sel = MATS++ ?
Alg_sel = MARCH X ?
Perform MATS Test on the Memory
Perform MATS+ Test on the Memory
Perform MATS++ Test on the Memory
Perform MARCH-X Test on the Memory
Alg_sel = MARCH B ?
Alg_sel = MARCH Y ?
Alg_sel = MARCH A ?
Alg_sel = MARCH C ?
Perform MARCH-B Test on the Memory
Perform MARCH-Y Test on the Memory
Perform MARCH-A Test on the Memory
Perform MARCH-C Test on the Memory
STOP

Fig 1: Flow chart showing MTC implementation

After designing and testing MTC (memory test controller), the next step is the design of control logic for repair module.

Memory test algorithms detect only the faults and determine whether the chip is faulty or not. In the case of repairable memories along with testing the location of faults and repair is also required. Including extra rows and columns that can be swapped for defective elements known as adding redundancy to the chip-can, in certain instances, help raise memory yield substantially. The test and repair logic modules should be present in the memory wrapper. Memories can have either spare rows, or spare columns or both. In some RAMs, the memories are arranged in block fashion with spare rows and columns. The arrangement is shown in Fig. 2.

The memory cell is tested using the test algorithms and taken the error addresses. The memory matrix is declared in the software which has the same number of cells as in actuality. The Error addresses are marked as 1's and rest as 0's in the memory matrix which we have declared in the software. It is as shown in the Fig 3.

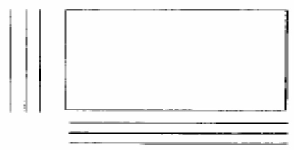

Fig 2 : Arrangement of memory with spare rows and columns

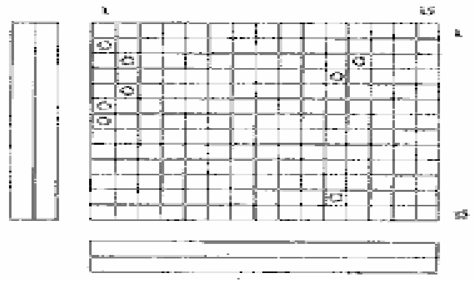

Fig 3 : Memory matrix indicating the error addresses

The program checks for the number of errors in each row and column in the memory block. The error in rows and columns are stored in the array and that array is sorted in ascending order for the maximum error addresses . Those addresses are sent to fuse box. If the spare rows and columns are 3 then first three addresses are sent to fuse box. In the fuse box the mapping will be done. The error row or column is replaced by the spare rows present in the memory. The flow chart for the control logic is as shown in Fig 5.3 and 5.4. The repair system is limited in the paper till the storage of addresses of maximum errors in the rows and columns which is equal to the spare rows. This finishes the control logic of the repair mechanism.In this procedure a file which has the error addresses is taken and then it is put in the matrix form to calculate the maximum errors in rows and columns. The procedure is given in the below algorithm.

After generating the addresses the addresses are given to the fuse box where the address mapping is done. One problem with certain redundancy models is that as the size and complexity of the SoC grows, adding extra rows and columns will become more and more burdensome, adding to the cost and intricacy of the chips. But in applications where accuracy is required this procedure is an advantage.

Algorithm for the implementation of the repair module:

1. Open the error address file and initialize an array equal to the memory size in the C program.
2. Make the array element of that error address equal to 1. Other array elements are initialized to 0.
3. Arrange it the matrix form of 1024X8 by matrix operation.
4. Make the count equal to number of redundant rows available in the memory
5. See the maximum number of errors(1's) present in the column. Display the column address
6. Replace that column by 0. Decrement count

7. Check if count is zero.If it is zero then go to step 5 or else go to step 8.
8. Initialize counter value to the number of redundant rows present in the memory.
9. See for maximum number of errors present in the row.
10. Display the row address and decrement the count.make that row as zero.
11. Check if row count is zero.If it is not zero then go to step 9. else go to next step.
12. Store all the row and column addresses which has the maximum errors

This is the control logic for the repair module which has been developed and checked in C. MARCH tests are extensively being used today for Functional testing of SRAM and DRAM technologies. They are more efficient then older classical pattern based tests with better fault coverage. This paper concentrates on producing IP to perform test and repair and diagnosis. This IP is designed to increase the reliability and performance.

IV. CONCLUSIONS:

MARCH tests are extensively being used today for Functional testing RAMs. They are more efficient then older classical pattern based tests with better fault coverage. The Memory Test Controller was successfully designed. The simulation was done using ModelSim software and it was implemented on the FPGA to check for the real time inputs. The hardware code is written in such a way that it can be modified for the any memory capacity by doing very minimal changes in the hardware. The repair module is done using C language which tells the addresses of the maximum number of errors in either row or column. One problem with certain redundancy models is that as the size and complexity of the SoC grows, adding extra rows and columns will become more and more burdensome, adding to the cost and intricacy of the chips. But in applications where accuracy is required this procedure is an advantage. The project was successfully designed and tested.

Test procedures that are considered in the project are widely used today for testing chip level, array level and board level functional memory defects**.** The project can be further improved by adding Neighborhood pattern sensitive tests and tests to check for all the linked errors completely in the memory test controller. This project is limited to the control logic design. The data path design is the next step in the project. This Project can also be converted to ASIC implementation if data path is designed and further above mentioned two tests are added.

V. REFERENCES:

[1]. Memory testing, Cheng-Wen Wu, Lab for reliable computing (LaRC), EE, NTHU.

[2]. Fault models and Memory Testing, Cheng-Wen Wu, Lab for realiable computing(LaRC), NTHU.

[3]. Using march tests to test SRAM'S, Van De Goor, A.J.; Design & test of Computers, IEEE Volume 10, Issue 1, march 1993 Page(s):8 – 14.

[4]. Essentials of Electronic Testing, Michael L.Bushnell, Vishwani D.Agarwal.

[5]. A.J. van de Goor, Testing Semiconductor Memories, Theory and Practice, John

Wiley&Sons, Chichester, UK, 1991.

[6]. S.M. Thatte and J.A. Abraham, 'Testing of Semiconductor random Access Memories, IEEE Computer Society Press, Los alamitos, Calif., June 1977, pp. 81-87.

[7]. R. Nair , "Comments on an optimal algorithm for testing stuck at faults in RAM", *lEEE Trans. Computers,* Vol. C-28, NO. 3, 1978, pp. 572-576.

[8]. D.S.Suk and Reddy "A march test for functional faults in semiconductor Random-Access Memories," *lEEE Trans on Computers,* Vol. C-30, No. 12, 1981, pp. 982-985.

[9]. M.S.Abadir and J.K. Reghbati, "Functional testing for semiconductor RAM" ACM computing surveys,Vol.15,NO.3, pp. 175-198

[10] http://www.eetimes.com

[11] SoC Yield Optimization via an Embedded-Memory Test by Samvel Shoukourian, Valery Vardanian, and Yervant Zorian published by IEEE CS and IEEE CAS.