International Journal For Technological Research In Engineering
Volume 2, Issue 2, October-2014 *ISSN (Online): 2347 - 4718*

SEMICONDUCTOR MEMORY TESTING ALGORITHM

Sagar Dokhale[1], Sunita Ugale[2]
E&TC Department, KKWIEER Nashik, India

ABSTRACT: In Semiconductor Memories Various faults occurs like stuck-at fault, coupling fault, Pattern Sensitive Fault, Bridging Fault. This paper explains a test algorithm to detect faulty row present in the memory. In this testing memory an algorithm based on Marching 1/0 is used. Xilinx ISE 9.1i is used for development of software algorithm. In this algorithm one test ram is created with verilog code which is instantiated in state machine. State machine will generate various data which is written to memory and read data from memory is also verified. If there is any mismatch between data written to the memory and data read from the memory the memory address is treated as faulty address and it can be given to the output.
Keywords: Buit-in Self Test (BIST), Bridging Fault (BF), Stuck-at Fault (SAF), Transition Fault (TF),

I. INTRODUCTION

The simplest test which detects SAFs, TFs and CFs are part of a family of tests called 'marches'. In order to detect all occurrences SAFs, TFs and CFs, all cells of memory should be able to be in state 0 and state 1; and for detection of TFs and CFs, all cells should have undergone ↑ and a ↓ transition. This implies that the march test is symmetric with respect to data values which are written and read.
Marching 1/0 algorithm: Its length is 14.n operations. Algorithm in march notation can be shown as below:

$$\{\Uparrow(w0);\Uparrow(r0,w1,r1);\Downarrow(r1,w0,r0);\Uparrow(w1);\Uparrow(r1,w0,r0);\Downarrow(r0,w1,r1)\}$$

M0 M1 M2 M3 M4 M5

It will be shown below marching 1/0 detects all AFs, SAFs and TFs. In addition it assumes that it will detect all CFs. The idea was that $\Uparrow$ march elements (M1 and M4) will finds CFs between the current cell the march element is operating upon and cells with a higher address, when the latter are read later on the same march element. Similarly M2 and M5 will find CFs for cells with address lower than the current cell. Below it is shown that Marching 1/0 is a complete test, thus all faults that should be covered are detected by the test.
1. Faults in the address decoder: The variable x can either be 0 or 1. x=0: $\Uparrow$ (r0,..,w1) and $\Downarrow$ (r1,...,w0).
This condition is satisfied by march elements M1 followed by M2. x=1: $\Uparrow$ (r1,..,w0) and $\Downarrow$ (r0,...,w1).
This condition is satisfied by march element M4 followed by M5.
2. Stuck-at faults: SAFs are covered, for example, with march element M1 each cell is read with an expected value of 0, then a 1 is written into the cell and the cell is read again; and with M4 each cell is read with an expected value of 1 , then a 0 is written into the cell and the cell is read again.
3. Transition Fault: < ↑ / 0> faults are detected by M1 and also by M5.These march elements generate an ↑ transition in every cell and immediately after cell is read < ↓ /1 > faults are detected by M2 and also by M4. Marching 1/0 is redundant because on M0, M1 and M2 are necessary.
M0 is necessary for the initialization of the memory cell. M1 is necessary for the detection of : < ↑ / 0> faults and for the detection of faults in the address decoder. M2 is necessary for the detection of < ↓ /1 > faults and for the detection of faults in the address decoder. State 0:In this stage Chip Select(CS) is made 1 and also Write Enable is made 1 so in next state specified data to be written in memory.

II. IMPLEMENTED FAULT MODEL

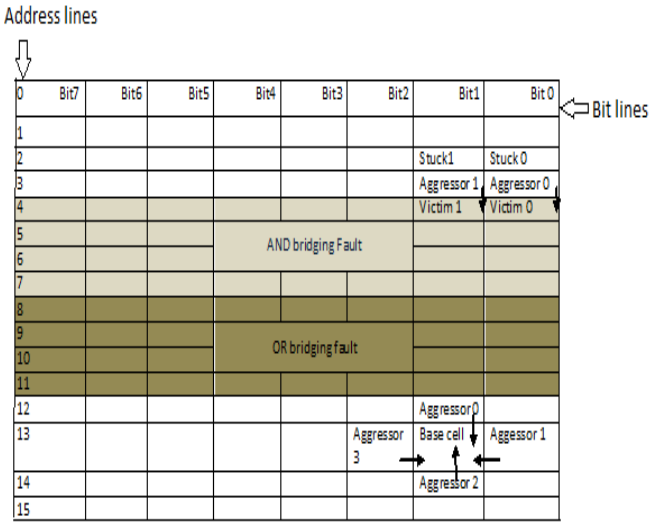

Fig.1 Implemented Fault model

I have designed 16 byte SRAM in Verilog HDL.I also introduced some faults in the memory as a first step of project. As shown in figure 1. I have introduced stuck at fault in row 2nd,coupling fault in row 3rd and 4th ,AND Bridging Fault(ABF) in row 4,5,6 and 7 and OR Bridging Fault in row 8,9,10 and 11.Pattern sensitive fault in row 13.Detailed discussion is done in Fig.1.

III. ALGORITHM USED FOR TESTING

State 1: In state 1 Data h'00 is written on memory cell.
State 2: In state 2 it is checked that whether address value is at the end of memory .If it is not at the end then data h'00 is written in a whole memory.
State 3: In state 3 Address value is made to 0 and moved to

step 4.
State 4: In state 4 it is checked that whether our memory consist of data is h'00 if it is not then fault reg is raised to 1 and given address is sent on faulty address output register. Again WE=1 and OE=0 for writing a data in a memory.
State 5: In state 5 data h'ff is written on one address and again data h'00 is read in
State 4: after control came to state 5 again data h'ff is written this continue till end of the memory is reached.
State 6: In state 6 data h'ff is read from bottom of memory and data h'00 is written on the same address after this address is decremented. OE and WE are enabled and disabled as per need in this state.
State 7: In state 7 data h'ff is written in ascending order on all memory addresses. When bottom of memory reach control is transferred to state 8.
State 8: In state 8 data is read in ascending order it is checked that whether data is h'00 or not. If bottom of memory is reached then control transferred to state 10.
State 9: In state 9 data to be written is set to h'00.OE=1 and WE=0 then control again transferred to state 8.
State 10: In this state it is checked that whether data is h'00 and if it is then WE is made 1 and OE is made 0 and control is transferred to state 11.If 0th address reached then control transferred to state 12.
State 11: In state 11 data h'ff is written into the memory in descending order and control alternatively transferred to state 10 and state 11.
State 12: In this state the state machine will remain in the same state.

IV. RESULTS

I have provided inputs to RAM by means of state machine which generates addresses as well as data for the memory. Same state machine does the work of comparison of written data and data read from the memory on mismatch of data address is given to faulty address output

Fig 2.Detection of faulty address rows 2 and 4

Fig 3. Detection of faulty address rows 7 and 11

Fig 4.Detection of faulty address 11 and 13

Fig.2 , Fig.3 and Fig.4 shows different faulty addresses in the memory. I have introduced various faults so the same address rows are detected as faulty by the algorithm. As seen from memory module I have introduced stuck at fault in second row ,coupling fault in row 4 , bridging fault in row 11 and pattern sensitive fault in row 13.So the faulty rows can be detected by this algorithm.

V. CONCLUSION

I have tested my algorithm for stuck at fault so there is 100 % coverage in detecting stuck at fault. Coupling fault is detected nearly 8 out of 10 times. Coverage for pattern sensitive fault is nearly 62%.By this modified march algorithm the data on the memory address rows can be written instead of bit by bit writing. Faulty address can be displayed on output LEDs.

REFERENCES

[1] A.J. van de Goor Testing Semiconductor Memories, chap 1; pp9-15, chap2; pp27-56, ComTex Publishing, Netherlands, 1998.
[2] E.Rob Dekker, Frans Beenker, Loek Thijssen, "Fault Modeling and Test Algorithm Development", in 1988 International Test Conference, IEEE.
[3] Heiko Ehrenberg, Chair, IEEE P1581 Working Group "IEEE P1581 can solve your board level memory cluster test problems" in International Test Conference, IEEE, 2007.
[4] Ad J.van de Goor, Magdy S. Abadir, Alan Carlin ."Minimal Test for Coupling Faults in Word-Oriented Memories" in Proceedings of the 2002 Design, Automation and Test in Europe Conference and Exhibition (DATE.02) IEEE.
[5] Kuo-Liang Cheng, Ming-Fu Tsai,and Cheng-WenWu, "Neighborhood Pattern-Sensitive Fault Testing and Diagnostics for Random-Access Memories", in "IEEE transactions on computer-aided design of Integrated Circuits and Systems", VOL. 21,NO. 11, NOVEMBER 2002.
[6] Zaid Al-Ars, A.J. van de Goor, "Transient Faults in DRAMs: Concept, Analysis and Impact on Tests" in IEEE, 2001.

