OPTIMIZING MARCH ALGORITHM FOR TESTING EMBEDDED MEMORY: A CASE STUDY

[1]Nitin Sharma, [2]Dr. Pradeep Kumar, [3]Mr. Sunil Kumar

ASET, AMITY UNIVERSITY, Noida (India)

ABSTRACT

The increase in integrity of the recent VLSI technology has enabled a trend of a lot of new, small and portable applications. The popularity of these portable applications, for ex notebook computers and cellular phones demand even higher performance and lower-power consumption. Embedded semiconductor memories form a big portion of the overall hardware on these portable devices. Testing the semiconductor memories is increasingly important today because of the high density of current memory chips.March algorithms are well known for memory testing because of their simplicity and good fault coverage. This is the reason that they are the dominant test algorithms implemented in most modern memory BIST. The paper describes the proposal to optimize one of the most popular march elements based algorithm c-. The proposed march algorithm is modified march c- algorithm which uses concurrent technique. The basic purpose of this proposal is to reduce the complexity of implementation and the test timey. Because of concurrency in testing the sequences the test results were observed in less time than the traditional March tests. This technique is applied for a memory of size 1024x16 and is generic enough to be extended to any other memory size.

Keywords : Embedded memory, fault coverage, March test

I INTRODUCTION

With every techno node and new product lines being attacked in semiconductor landscape, due to ever growing sizes and density, Embedded Memories are using more and more complex design structures resulting in the chances of occurring manufacturing defects to be more compared to any other embedded core on SOC. Hence testing of embedded memory is becoming an ever growing real challenge for every design architect. For SOCs,to have direct access tothe core is already one of the major problems in the field of testing and diagnosis [1-4]. Further because of the limited available bandwidth between the primary inputs of the system chip and the embedded core, the external access for test purpose is becoming increasingly infeasible. This has prompted a very strong interest in the self-test of the embedded memory arrays. In particular, mostly because they provide defined detection properties for classical memory array faults such as stuck at faults and transition faults, functional March tests have found wide acceptance in this domain.

During Memory tests, each location in a memory device is tested to be working. This testing involveswriting a specific set / pattern of data to each memory address location and verifying this data by reading it back and

comparing with an already anticipated response. Depending on the values readback, if they are the same as those that were written, the memory device is said to pass the test, otherwise devicefails. There are various test methodologies that have evolved from the years to identify the memory defects.One suchtest methodology is memory built in self-test which involves built in self-test circuitry for each memory array to apply some data pattern in a pre-defined manner, reading back the response, comparing the response and reporting the status. The memory BIST is implemented embedding various march algorithms to test the memories.

The advantage of March tests, which makes it acceptable from industrial point of view, lay in the fact that high fault coverage can be obtained and the testtime were usually linear with the size of the memory.March based algorithms were capable of locating and identifying the fault types ultimately helping to catchdesign and manufacturing errors.

In order to further improve this acceptance of march based tests, the method proposed in this paper is Modified March C- algorithm with concurrent technique. The intent is of this modified algorithm is to retain the high fault coverage of March C but the test run should finish ina reduced time frame or reduced number of cycles. The paperdescribes the functional fault models in the memory, classical and March based tests in section II. Theproposed Modified March c- algorithm is discussed in section III. Results and comparisons are discussedin section IV. Section V concludes the work.

II HISTORY OF FUNCTIONAL FAULT MODELS

For testing purpose,various functional faults are represented as fault models modeled after observing the behavior of faults in memories so thatfunctional tests to detect these faults can be used. This modeling helps to clarify, simplify and generalize thetesting approach of a memory. The fault model of the faults determines the quality of tests strongly in terms of its fault coverage, its test vector length as well as the test time.

There are various fault models to test the functional faults. Some most popular ones are such as stuck at faults.While dealing with SRAMS, coupling faults are also considered. Other faults to consider are Address decoder faults and bridging faults when dealing with DRAM. The most commonly occurring faults in general are stuck at faults [5-7].

Stuck at fault (SAF): The stuck-at fault (SAF) considers that the logic value of a cell or line isalways 0 (stuck-at 0 or SA0) or always 1 (stuck-at 1 or SA1). To detect and locate all stuck-at faults, a testmust satisfy the following requirement: from each cell, a 0 and a 1 must be read.

Transition Faults(TF): The transition fault (TF) is a special case of the SAF. A cell or line that failsto undergo a 0 → 1 transition after a write operation is said to contain an up transition fault. Similarly, adown transition fault indicates the failure of making 1 → 0 transitions. According to van de Goor [8, 9], a testto detect and locate all the transition faults should satisfy the following requirement: each cell must undergoa↑ transition (cell goes from 0 to 1) and a ↓ transition (cell goes from 1 to 0) and be read after eachtransition before undergoing any further transitions.

The algorithms targeting fault detection for both SAFs and TFs are MATS++ andMarch C- algorithm.Now with the new modified C- Algorithm, although different in test length, there is no fault coverage loss compared to MATS++ and traditional March C-for capturing stuck at and transition fault. In additions the fault coverage in context of detecting other faults is as good as well. The detection process can be understood by examining theMach C- algorithm as indicated in expression below.

March Test Notation: A March test consists of a finite sequence of March elements [10-12]. AMarch element is a finite sequence of operations or primitives applied to every memory cell beforeproceeding to next cell. For example, ↓(r1, w0) is a March element and r0 is a March primitive. The addressorder in a March element can be increasing (↑), decreasing (↓), or either increasing or decreasing (↕). Anoperation can be either writing a 0 or 1 into a cell (w0 or w1), or reading a 0 or 1 from a cell (r0 or r1).Accordingly notation of March C- test is described as follows:

{↕(w0);↑(r0,w1);↑(r1,w0);↓(r0,w1);↓(r1,w0);↕(r0)}

↓ ↓ ↓ ↓ ↓ ↓

$M_I M_{II} M_{III} M_{IV} M_V M_{VI}$

March C- algorithm has 6 elements as shown with a complexity of 10n.

III MODIFIED MARCH C- ALGORITHM

The proposed Modified March C- algorithm almost similar to March C- but it follows concurrencyin testing the sequences. The steps for following the concurrency are as follows:

- Group entire memory into subgroups.
- For each subgroup, generate all test vectors for the first fault in the group.
- Simulate all faults in the subgroup to select one vector that detects most faults in thatsubgroup. If more vectors than one detect the same number of faults within the group, then select the vector that detects most faults outside the group as well.
- Apply the final test vectors to all subgroups concurrently

In the proposed method the memory is divided in to two subgroups such as M1 and M2. Then,the algorithmsare applied for concurrency. The following are the elements in Modified March C- algorithm.

M1:{↑(w0);↑(r0,w1);↑(r1);↓(w0);↓(r0,w1);↓(r1);

M2: {↑(w1);↑(r1,w0);↑(r0);↓(w1);↓(r1,w0);↓(r0);

The number of March elements is same as traditional March c- and is 6 but because of concurrency thecomplexity is reduced to 8n.

The pseudo code for modified march c- is as follows:

```
//for writing 0s in block 1 and writing 1s in block 2, let n and m are rows and columns
for(i=0;i<(n-1)/2;i=i+1)
begin
for(j=0;j<(m-1);j=j+1)
mem[i][j]=0;//write 0 in m1
end
for(i=(n-1)/2;i<(n-1);i=i+1)
begin
for(j=0;j<(m-1);j=j+1)
mem[i][j]=1;//write 1 in m2
end
```

```
//for reading background and for writing alternatively
for(i=0;i<(n-1)/2;i=i+1)
begin
for(j=0;j<(m-1);j=j+1)
begin
if(mem[i][j]==0)
mem[i][j]=1;
else return;
end
end
for(i=(n-1)/2;i<(n-1);i=i+1)
begin
for(j=0;j<(m-1);j=j+1)
begin
if(mem[i][j]==1)
mem[i][j]=0;
else return;
end
end
```

According to Modified March C- elements, when 0s are written in one memory group, 1s will be written in another group concurrently. So the test sequence can be taken through an inverter hence true formwill goes to one block of memory and complement form will goes to another block of memory. The methoddirectly reduces the time required to write and read the bit concurrently. Ultimately this reduces the logic to be put on the address decoder path and operation controller path of the memory BIST circuit there by resulting in lesser logic on the path to memory. This reduces the test time and testcosts also. Finally, there may be additional design cost in terms of some hardware gates added to generate the test sequence to perform the test concurrently.

IV RESULTS AND COMPARISONS

Table 1 indicates delay introduced by performance for each element present in traditional March C- algorithm under faulty condition. Using SA fault models the overalldelay observed as 11.602ns.

Table 2 shows the delay performance using Modified March C- algorithm. In this also delayperformance were calculated under faulty conditions. Under faulty condition theoverall delay was observed as 11.494ns. Hence it is proved that using Modified March C- algorithm usingconcurrency the overall delay is greatly reducing. It is giving at speed test performance than any othertraditional algorithm. The result tables also provide the information on minimum input arrival time beforeclock and maximum output time after the clock. Simulation was carried using Xilinx ISE 10.1i tool for the

device XC3S4004tq144 and tested on Spartan 3 kit. Fig 1 and 2 shows the simulation results respectively formodified march elements I and II when fault is imposed.

Table 1: Result of traditional March C- Algorithm

March Element	Min period	Min i/p arrival time before clock	Min o/p required time after clock
M_I: ↕(w0);	1.483	3.439	6.314
M_{II}: ↑(r0,w1);	1.585	3.504	6.318
M_{III}: ↑(r1,w0);	1.585	3.504	6.318
M_{IV}: ↓(r0,w1);	1.585	3.504	6.318
M_V: ↓(r1,w0);	1.585	3.504	6.318
M_{VI}: ↕(r0)	2.196	3.955	6.318

All the delay times mentioned are in ns.

Table 2: Result of traditional March C- Algorithm

March Element	Min period	Min i/p arrival time before clock	Min o/p required time after clock
M_1: ↑(w0);	1.483	3.439	6.28
M_2: ↑(w1);			
M_1: ↑(r0,w1);	2.132	4.755	6.441
M_2: ↑(r1,w0);			
M_I: ↑(r1);	2.132	3.96	6.318
M_2: ↑(r0);			
M_I:↑(w1);	1.483	3.439	6.28
M_2:↑(w0);			
M_I: ↓(r0,w1);	2.132	4.755	6.441
M2: ↓(r1,w0);			
M_I:↓(r0)	2.132	3.96	6.314
M_2:↓(10)			

Table 3: Comparison

Type of Algo	Complexity	Delay
Traditional C-	10N	13.782
Modified C-	8N	11.783

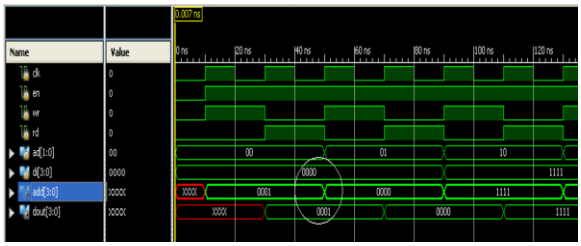

Figure 1.

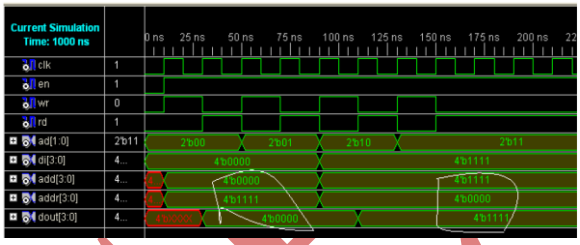

Figure 2.

Figure 1. Simulation results for modified march C- element I {M1: ↑ (w0)}{M2: ↑ (w1)}

Figure 2. Simulation results for modified march C- element II {M1: ↑ (r0,w1)}{M2: ↑ (r1,w0)}

V CONCLUSION

This paper defines the functional fault model and compared the traditional march c- algorithm withmodified march c- algorithm in terms of speed of the test sequence and complexity of the number of testsequences. The crucial part in testing is how well the test can be completed in minimum time with minimaltest length. The modified march algorithm has proved that the test length is minimal as well as the timerequired to test SAF also minimum when compared with traditional march c-. Hence this modified march cismuch comparable and could be used for detection of various faults other than SAF as future work.

REFERENCES

[1] Shibaji Banerjee, Dipanwita Roy Chowdhury and Bhargab B. Bhattacharya, in proceedings of the 2005 IEEE International Workshop on Memory Technology, Desgin and Testing (MTDT'05).

[2] The national Roadmap for Semiconductors, 2000. SemiconductorIndustry Association.

[3] E Eric Dupont, Michael Nicolaidis, and Peter Rohr, " Embedded Robustness IPs for Transient Error free ICs", IEEE Design & Test of Computers, 2002.

[4] Dupont, E. Nicolaidis, M. And Rohr .P. " Embedded Robustness IPs for Transient Error Free ICs" IEEE Design &Test of Computers, Vol 19, No 3, pp 56-70, May-June, 2002.

[5] J.F.Li, K.L.Cheng, C.T.Huang, and C.W.Wu, "March based RAM diagnostic algorithms for stuck-at and couplingfaults", Proceedings IEEE ITC pp,758-767,2001

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] R. Dekker, F. Beenaker, L.Thijssen, "A realistic fault model and test algorithm for static random access memories", IEEE Trans. Computer-Aided Design, vol,9,99.567-572, june 1990.

[8] J. van de Goor. Testing Semiconductor Memories: Theory and Practice, A.J.vandeGoor,1998.

[9] Verilog Digital System Desigh, 2nd Edition, ZainalabedinNavabi, Tata McGraw Hill, 2008

[10] A.J.Van de Goor, Testing Semiconductor Memories, Theory and Practice, John Wiley & Sons, Chichester, England,1991.

[11] Van De Goor, A.J. Sehanstra. I., "Address and Data Scrambling: Causes and Impact on Memory Tests", IEEEDELTA Workshop, Christchurch, January 2002.

[12] ER.ManojArora, Er.ShipraTripathi, "Comparative Simulation of MBIST using March Test Algorithms",International Journal of Scientific & Engineering Research, Volume 2, Issue 12, December-2011