International Journal of Engineering Research in Electronic and Communication Engineering (IJERECE) Vol 3, Issue 5, May 2016

Basic March-C Algorithm based BIST for Embedded Memories in FPGA

[1]Hema G D, [2]DivyaPrabha, [3]M.Z. Kurian
PG Student (VLSI and ES) Dept. of ECE, Sri Siddhartha Institute of Technology,Tumakuru, Karnataka, India.
Associate Professor, Dept. of ECE, Sri Siddhartha Institute of Technology,Tumakuru, Karnataka, India.
HOD, Dept. of ECE, Sri Siddhartha Institute of Technology,Tumakuru, Karnataka, India.
d.hema71@gmail.com, dpssit@gmail.com

***Abstract*— Built in Self Test is one of the widely used methods for memory testing and it is the cost effective method. The fault in the memory is due to the complexity of the design rules. For complex applications, the memories without faults are necessary. There are many test algorithms for testing of memories, march based tests are the dominant testing algorithms due to simplicity and ability to test the faults. Because of this, march tests are implemented in most modern memories BIST. In this project, by considering optimized march-c algorithm to test the faults. This algorithm uses the concurrent technique. Because of concurrency the testing time is reduced compared to basic march c algorithm. This technique is applied to 256x8 memories it can be extended to any size. For the effectiveness of this algorithm, Built-in self-test technique is considered to test embedded memory of the FPGA.**

***Key Words:* BIST, March-C, Optimized March-C, FPGA.**

I. INTRODUCTION

Several Field Programmable Gate Array (FGPA) vendors have complicated memories within their devices. Xilinx, the Alter a and AT40k vendors are the examples of FPGAs. These memories are highly programmable. Because of a complexity of design, there is a chance of occurring manufacturing faults, because of this testing of the memory is required. The FPGA without fault is important for all applications**.** In FPGAs, the chances of occurring faults include memory resources, logic blocks, or inter-connects. These defects can be modeled as single and multi-cell memory faults. The dominant use of embedded memory cores along with emerging new architectures and technologies make providing a low cost test solution for these on chip memories a very challenging task.

Built in self-test is considered in the algorithm for testing of embedded memory. The memory is modelled in equivalent gate-level and run usingISIM simulation tool. Further, the FPGA implementation is done as well.

The usage of embedded memories is increased day by day; different memories are available in different sizes. To reduce the Size of the memory, by using complex designs. Hence, the manufacturing difficulty in producing the memory also increases. This requires the testing methods to test memories during manufacturing. The difficult task for design architect is to test memory. There are many methods are available for testing, march tests are the one of the main testing algorithm in use. The main faults occurs in the memory are stuck at fault and transition fault. The idea behind testing is that writing some set of data to the particular memory location and read back the value from the same location. If the written data and data obtained after testing are same then that indicates there is no fault in memory, if both are different there is a fault in memory.

Most of the memory testing works uses the basic March-c algorithm. But this method is not suitable for large memories. Disadvantages in existing system is time of testing is more, and separate BIST is required for each subgroup of memory.

The objective is to create both basic march-c algorithm architecture and optimized March-c algorithm based BIST architecture to test the memory inside the FPGA and comparing both in terms of time, area and speed. Finally, finding the capable architecture which tests the memory in less time.

This system proposes the architecture to test the memories using optimized March-c algorithm. It requires less time to test the memory compared to other architectures. Advantages of the proposed system is, it uses concurrency in testingand applicable for large memories. Single BIST is sufficient for a group of memories.

II. GENERIC BIST ARCHITECTURE

Test Pattern Generators (TPGs)-generates test pattern to the circuit under test.
Output Response Analyzer (ORAs)- evaluate the result of cut and required result.
Circuit Under Test (CUT) is the device used to test for faults (Memory).
Distribution system (DIST)-sends the data between TPGs to CUT and CUT to ORAs.
Interconnections (wires), busses, multiplexers and scan paths.
BIST controller is used in testing mode to control the BIST circuitry and CUT.
Test Response Analysis (TRA): It analyses the value sequence on PO and compares it with the expected output, based on the value compacted signature is also stored for future reference.

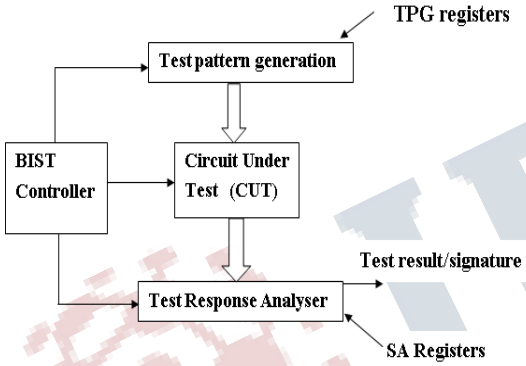

Fig. 1. Generic BIST Architecture

III. TEST ALGORITHMS

Name	Algorithm
MATS	{↕(w0);↕(r0,w1);↕(r1)}
MATS+	{↕(w0);↑ (r0,w1);↓(r1,w0)}
MATS++	{ ↕ (w0);↑ (r0,w1); ↓ (r1,w0, r0)}
MARCH X	{ ↕ (w0);↑ (r0,w1); ↓ (r1,w0);↕(r0)}
MARCH A	{ ↕ (w0); ↑(r0,w1,w0,w1); ↑(r1 ,w0,w1);↕(r1,w0,w1,w0); ↑ (r0,w1,w0)}
MARCH Y	{ ↕ (w0); ↑ (r0,w1, r1); ↓ (r1,w0, r0);↕(r0)}
MARCH B	{ ↕ (w0);↑ (r0,w1, r1,w0, r0,w1); ↑ (r1,w0,w1); ↓ (r1,w0,w1,w0);↓ (r0,w1,w0)}

Fig.2., Test Algorithms

A test algorithm is a finite sequence of test elements. A test element contains a number of memory operations, data pattern specified for the read operation, address specified for the read and write operations. We divided these test algorithms into two groups: Traditional tests and March-based tests. The following are the well known traditional tests: zero one, checkerboard, GALPAT and walking 1/0, sliding diagonal and butterfly. These are either simple, fast but have poor fault coverage or have good fault coverage but complex and slow. Due to these imbalanced conflicting traits, the popularity of these algorithms is decreasing. A March-based test algorithm is a finite sequence of March elements. A March element is specified by an address order and a number of reads and writes. The different types of march based tests are MATS, MATS+, March C-, March Y, March A, March B, and etc. Since March-based tests are all simple and flexible to possess good fault coverage, and they are dominant test algorithms implemented in most of the modern memory BIST.

A. MARCH C Algorithm

March test algorithm is a finite sequence of March elements. March element is specified by an address order and number of read/write operations. This section discusses March tests that are of $O(N)$ complexity. March tests are named so, because starting with the first memory location a 1 (or a 0) is written while locations previous to that keep their written 1 (or 0) values. So it appears like 1s (or 0s) are marching in from location 0 to the last location in the memory. This notation unambiguously specifies the testing procedure, and the number of reads and writes are easily seen that determine the order of a test procedure.

The March C algorithm is shown in below.
March C:{↕(w0);↑ (r0,w1);↑(r1,w0);↓ (r0,w1);↓ (r1,w0); ↕(r0)}.

Steps in March C- Test:

- In any order write 0s to all cells. (M0: ↕ (*w*0)).
- Read from the lowest address (expected read value is 0)repeat until the highest address is reached, and write a 1 at this address. (M1: ↑ (*r*0, *w*1)).
- Read from the lowest address (expected read value is 1), write a 0 at this address and repeat until the highest address is reached (M2: ↑ (*r*1, *w*0)).
- Read from the highest address (expected read value is 0), write a 1 at this address, and repeat until the lowest address is reached (M3: ↓ (*r*0, *w*1)).
- Read from the highest address (expected read value is 1), write a 0 at this address, and repeat until the lowest address is reached (M4: ↓ (*r*1, *w0*)).

- In any order (expected read value is 0) read from all the cell. (M5: ↕ (*r*0)).

This is the pseudo code for basic march c algorithm

```
always@(posedge clock)
begin
        if(reset) addra<= 0;
        else if(rst_addr_cntr) addra<= 0;
        else if(en_addr_cntr) addra<= addra + 1;
        elseaddra<= addra;
end


always@(posedge clock)
begin
        if(reset) dina<= 0;
        else if(rst_data_cntr) dina<= 0;
        else if(en_data_cntr) dina<= dina + 1;
        else dina<= dina;
end
```

B. Optimized March C Algorithm

The proposed Optimized March C algorithm almost similar to March C but it follows concurrency in 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 that subgroup. 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

The following are the elements in Optimized 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 the complexity is reduced.

According to Optimized 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 form will goes to one block of memory and complement form will goes to another block of memory. Hence the test sequence generator requires additionally one inverter in order to perform test concurrently. The method directly reduces the time required to write and read the bit concurrently. This reduces the test time and test costs also. Finally, there may be additional design cost in terms of inverter only which need to generate complement test sequence to other part of the memory block.

IV. HDL SYNTHESIS REPORT March C ALGORITHM

```
# Counters                 : 3
32-bit up counter          : 1
8-bit up counter           : 2
# Registers                : 6
1-bit register             : 2
4-bit register             : 1
8-bit register             : 3
# Comparators              : 1
```

Device utilization summary

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	78	11,776	1%
Number of 4 input LUTs	57	11,776	1%
Number of occupied Slices	64	5,888	1%
Number of Slices containing only related logic	64	64	100%
Number of Slices containing unrelated logic	0	64	0%
Total Number of 4 input LUTs	102	11,776	1%
Number used as logic	57		
Number used as a route-thru	45		
Number of bonded IOBs	5	372	1%
Number of BUFGMUXs	1	24	4%
Number of RAMB16BWEs	1	20	5%
Average Fanout of Non-Clock Nets	2.92		

Fig. 3., Synthesis report of basic march c algorithm

```
Timing Summary:
---------------
Speed Grade: -5

Minimum period: 4.455ns (Maximum Frequency: 224.459MHz)
Minimum input arrival time before clock: 3.215ns
Maximum output required time after clock: No path found
Maximum combinational path delay: No path found
```

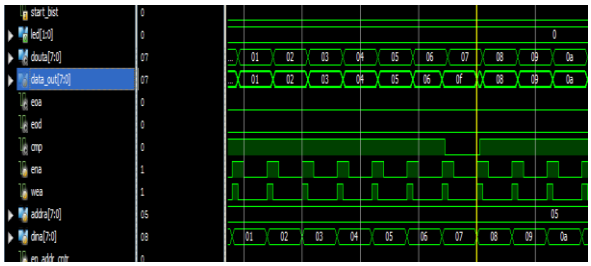

Fig.4., stuck-at 1 fault for basic march c algorithm

V. CONCLUSIONS

As the weight of embedded memory in aggressive System-on-Chips (SoCs) gradually increases, the importance of testing embedded memory in SoC increases. The most prevalent method of testing embedded memory is Built-in Self-test (BIST). BIST is the best solution for testing embedded memories within SOCs. It offers a simple and low cost means without significantly impacting performance. Here by considering the march C algorithm we find the time required for finding the faults. In further implementation by considering optimized march C algorithm has to prove the test length is minimal as well as the time required to test SAF also minimum when compared with traditional march c algorithm.

REFERENCES

[1] Modified March C - Algorithm for Embedded Memory Testing by muddapu parvathi, N.vasantha, K.Satya Parasad International Journal of Electrical and Computer Engineering (IJECE) Vol. 2, No.5, October 2012, pp. 571~576ISSN: 2088-8708.

[2] Der-Cheng Huang; Wen-Ben Jone, "A parallel built-in self-diagnostic method for embedded memory arrays," in Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.21, no.4, pp.449-465, Apr 2002.

[3] Design of Built-in Self-Test Core for SRAM by Reeja J. and Anusree L. S. International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Vol. 3 Issue 3, March – 2014.

[4] A Failure Testing System with March C-Algorithm for SingleEvent Upset by Peng Wang, ZhenLi, Chengxiang Jiang, Wei Shao and Qiannan Xue International Journal of Hybrid Information Technology Vol.7, No.2 (2014), pp.95-102.

[5] Design of Improved Built-In-Self-Test Algorithm (8n) for Single Port Memory by Manoj Vishnoi, Arun Kumar, Minakshi Sanadhya International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-5, November 2012.

[6] J.vande Goor.Testing Semiconductor Memories: Theory and Practice. A.J.vande Goor, 1998.

[7] Testing of Embedded System Version 2 EE IIT, Kharagpur Lesson 40 Built-In-Self-Test (BIST) for Embedded Systems.