Design of Built-in Self-Test Core for SRAM

Reeja J.
PG Scholar, VLSI & Embedded Systems, ECE Department
TKM Institute of Technology
Karuvelil P.O, Kollam, Kerala-691505, India

Anusree L. S.
Assistant professor, ECE Department
TKM Institute of Technology
Karuvelil P.O, Kollam, Kerala-691505, India

***Abstract*—Nowadays, usage of embedded memories is more than half of the die area for a typical SoC. Due to the complexity of memory architectures, the possibility of occurring manufacturing defects is more. Hence, memory testing is a very challenging task. Built-in self-test (BIST) has been proven to be one of the most cost-effective and widely used solutions for memory testing. It is used to confirm that each location in a memory device is working. In this project, a BIST core for testing SRAM using March C- algorithm has been proposed. The BIST core has been designed to test 8 bit SRAM using March C- test patterns. The generated test patterns to be applied to each memory address and the test response can be evaluated by an output response comparator. Different fault models can be detected by Bit-Oriented and Word-Oriented March C- algorithms. The main advantage of March C- algorithm is high fault coverage and linearity of the test time with memory size. The design architecture can be written using VHDL code and simulated using Xilinx ISE tools and ModelSim.**

Keywords— BIST, March Test Algorithm, Memory fault models

I. INTRODUCTION

As process technologies continue to shrink and memory size and design complexity grow, it has become increasingly difficult to achieve high manufacturing yield. Embedded memories are the most dense components within a system-on-chip (SoC), accounting for more than 50 percent of the chip area. Implemented using aggressive design rules, embedded memories tend to be more prone to manufacturing defects and field reliability problems than any other core on the chip. Applications that require lots of memory are served by designs that embed large numbers of memory bits per chip, creating more powerful SoCs, but this has the associated problems of increased die size and poor yield. As design applications require more memory, it is essential to implement a comprehensive embedded memory test help to achieve high yield. BIST is considered as one of the most promising solution for memory testing. The basic idea of BIST, in its most simple form, is to design a circuit so that the circuit can test itself and determine whether it is "good" or "bad" (fault-free or faulty, respectively). This typically requires that additional circuitry and functionality be incorporated into the design of the circuit to facilitate the self-testing feature. This additional functionality must be capable of generating test patterns as well as providing a mechanism to determine if the output responses of the circuit under test (CUT) to the test patterns correspond to that of a fault-free circuit. The main feature of the BIST is the capability to test deeply the memory through an in- built algorithm. Basically the testing algorithms can be divided into two types: Traditional test algorithms and March based test algorithms. Traditional test are checkerboard, GALPAT, Walking 1/0, butterfly and etc. March algorithms are preferred over Traditional testing algorithms because March algorithms are highly linear, simple and good fault coverage. Among March test algorithms, the March algorithm is practical to offer the highest fault coverage. So March C- algorithm is considered here for the design of BIST for SRAM.

The report is organized as follows. Chapter 2 includes the literature survey of the project. It gives the basic idea of Memory BIST , SRAM memory fault models and March C- algorithm. Chapter 3 presents the design of Bit-Oriented March C- algorithm and the architecture of March C- Memory BIST for 8-bit SRAM. The simulation results are given in the Chapter 4. It includes the simulation result obtained from ModelSim and finally Chapter 5 concludes this project and outlines the further researches followed by the references.

II. LITERATURE REVIEW

A. Built-In Self-Test

Built-in Self Test, or BIST, is the technique of designing additional hardware and software features into integrated circuits to allow them to perform self-testing, thereby reducing dependence on an external automated test equipment (ATE). The general BIST architecture is shown in Fig 1.1.

The general BIST architecture consists of mainly four blocks. They are,

1. BIST test controller, which controls the BIST circuit.
2. Test generator, which controls the test address sequence.
3. Response verification as a comparator, which compares the memory output response with the expected correct data.
4. Circuit Under Test

A more recent method of memory testing is to include the tester function right in the silicon by placing an address generator, data generator and algorithmic sequencer into the silicon design.

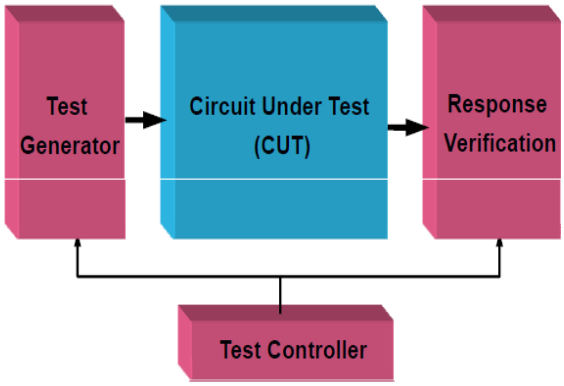

Fig.1.General BIST Architecture

The Memory BIST approach is commonly used because it provides efficient, less costly and speed testing solution for memory. The memory BIST provides several advantages such as high fault coverage, high speed testing, low area and low cost than other testing methods. The FSM based memory BIST having advantage of speed testing, small area and compact but less flexibility.

B. Memory Fault Models

In general, there are four memory fault models:

i) Stuck At Faults
ii) Transition Faults
iii) Address Decoder Faults
iv) Coupling Faults

i) Stuck At Faults Model (SAF)

A stuck at fault implies that a write operation is unable either a 0 - 1 or a 1 – 0 transition in the memory cell where the fault exists. It indicates that a cell is locked in one state or another. A single cell which is defect free can be written to either state and when read retains the information that was originally in the cell. There are two main types are stuck at 1 or stuck at 0. For example stuck at 0 in a memory; no matter what happens in the cell, whether it is read or written, regardless of intended data value, the cell stays at a "0".

ii) Transition Faults Model (TF)

The transition fault is memory cell will retain either state. However once it is written to one state it cannot transition back. Thus, when the memory is powered up the cell may be in either a "0" or a "1" state. It can only written in one direction where it is possible to transition this cell from a "0" state to a "1" state but it cannot transition back. Transition faults are two types: Up Transition Fault and Down Transition Fault.

iii) Address Decoder Faults (AFs)

Address decoder faults (AFs) are faults in the address decoder. There are four types of address decoder faults.

1) With a certain address, no cell will be accessed.
2) A certain cell will not be accessible.
3) With a certain address, multiple cells are accessed simultaneously.
4) A certain cell can be accessed with multiple address.

iv) Coupling Fault Model (CF)

Coupling fault occurred because of the regularity of its structure, a memory chip may experience a change in one cell due to an intended change in another cell. The coupling is due to shorts or parasitic effects such as stray capacitance. There are two CF types: Inversion and Idempotent coupling Fault.

a) Inversion Coupling Faults (CFin)

The coupling between two adjacent cells "i" and "j"; the faults sensitized by transition write operation (write 0 - 1 or 1 - 0) to cell "j". Cell "j" is called the coupling cell and inverts the contents of cell "i" which is known as the coupled cell. For example: transition write operation 0 - 1 to cell "j" causes a 1 - 0 inversion coupling fault in cell "i".

b) Idempotent Coupling Faults (CFid)

An idempotent coupling fault (CFid) is where an increse (↑) or decrease (↓) transition in cell sets cell to 0 or 1. This is denoted as <↑; 0> or <↑; 1> depending on whether cell i is set to 0 or 1, for a rising transition for cell j. The other two idempotent coupling faults are <↓; 0> and <↓; 1>.These faults also involve coupling between two adjacent cells "i" and "j", and are sensitized by a transition write operation. The difference between inversion and idempotent is that the transitions write operation to cell "j" forces the contents of cell "i" to a fixed value which is not necessarily the inverse of the value written to cell "j".

C. March Test 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. For ease of explanation of March tests, a notation has been devised by Van De Goor [9] a subset of which is shown in Table 2.1. 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.

Table.2.1 Test notations

R	Memory Read operation
r0	Read a 0 from the memory
r1	Read a 1 from the memory
W	Memory Write Operation
w0	Write a 0 to the memory
w1	Write a 1 to the memory
↕	Increasing memory address ordering
↑	Decreasing memory address ordering
↓	There is no difference between different addressing orders.

There are a few of test algorithms for SRAM testing that are rather simple for BIST implementations include variations of the Modified Algorithmic Test Sequence (MATS) and

March tests including the March Y algorithm that used for the FSM based Test Pattern Generation. Some of the March based tests are MATS, MATS+, MATS++ March X, March Y, and March C-.For highest fault coverage, March C- algorithm is used as the memory test algorithm.
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:
1. Write 0s to all cells in any order (M0: ↕ (*w*0)).
2. Read from the lowest address (expected read value is 0), write a 1 at this address and repeat until the highest address is reached (M1: ↑ (*r*0, *w*1)).
3. 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)).
4. 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)).
5. 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*)).
6. Read from all cells in any order (expected read value is 0) (M5: ↕ (*r*0)).

In Fig.2., the State diagram of March C- algorithm, first state denotes initial state, intermediate states (M0, M1, M2, M3, M4, M5) denotes March elements of March C- algorithm and last state denotes final state.

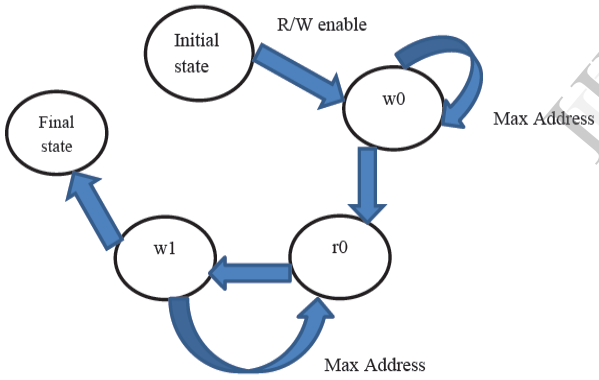

Fig.2. State diagram for first two operations of March C -algorithm

III. Proposed Method

A. System architecture

FSM based Bit-Oriented Memory Test algorithm is designed in this project. Memory test algorithms are used to confirm that each location in a memory device is working. This involves writing a set of data to each memory address and verifying this data by reading it back. If all the values read back are the same as those that were written, then the memory is fault free, otherwise faulty. For highest fault coverage, March C- algorithm is used as the memory test algorithm [4].

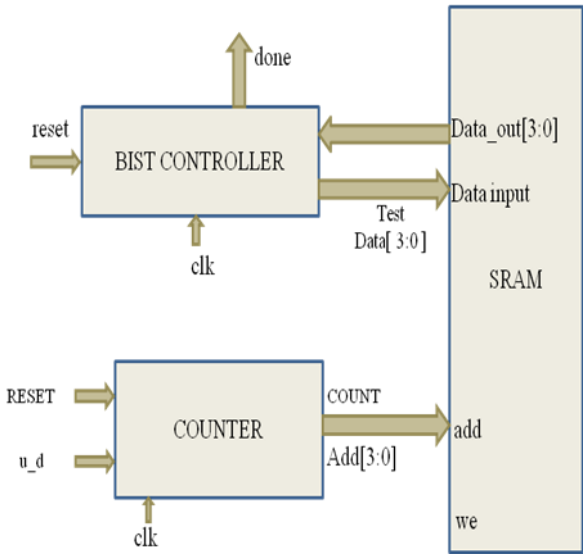

Fig.3. Block Diagram of Bit Oriented March C- Algorithm

The test begins starting with the Test controller that gives input to the SRAM. An FSM works on the Test controller which starts from the first March element M0 to March element M5. Increment / Decrement operation is done by the up-down counter. When up signal is high, it starts to count from lower order address to higher order address. When up signal is low, it starts to count from higher order address to lower order address. Counter output is given to the read/write address of SRAM. After completion of performing all March element of March C- algorithm, the Test controller block makes the output signal (done) high.

Fig.4. shows the MBIST architecture. The memory to be tested is shown in gray, and solid line blocks show the test circuitry. Test data that are to be applied are generated by this MBIST circuitry and applied to the memory. As data are being read from the memory, they are compared with the reproduction of the same data that was written into specific memory locations. After writing and reading all locations, expect all data read from the memory to be the same as those that were written into it. Counter-sequencer handles the ten phases of March C- test. Here, memory that is to be tested , has a word-length of eight bits.

i) *March C- BIST Counter-sequencer*

It is a 13 bit counter, which provides address, direction of address generation (from highest to lowest or visa versa), test data, and switching between reading and writing the memory. The six least significant bits of the counter-sequencer provide addressing for all locations of the memory. The next four bits to the left of the address group of bits specify the ten consecutive read/write operations of March C-. Also these four bits along with the three most significant bits of the counter-sequencer are decoded to generate the appropriate test vectors for testing the memory words. According to March C- algorithm, in Steps 1 and 6, the address increments (or decrements) and, for the specific address, only read or write operation will take place. In all other steps, both read and write operations must be done before going to the next address. To implement this, when the

counter-sequencer is in Steps 1 or 6, it behaves normally and increments or decrements by 1 with every clock cycle. In all other steps, the address of memory (the six least significant bits (cnt_ reg[5:0]) do not change for two consecutive cycles, but cnt_reg[9:6] increments and then decrements by 1.

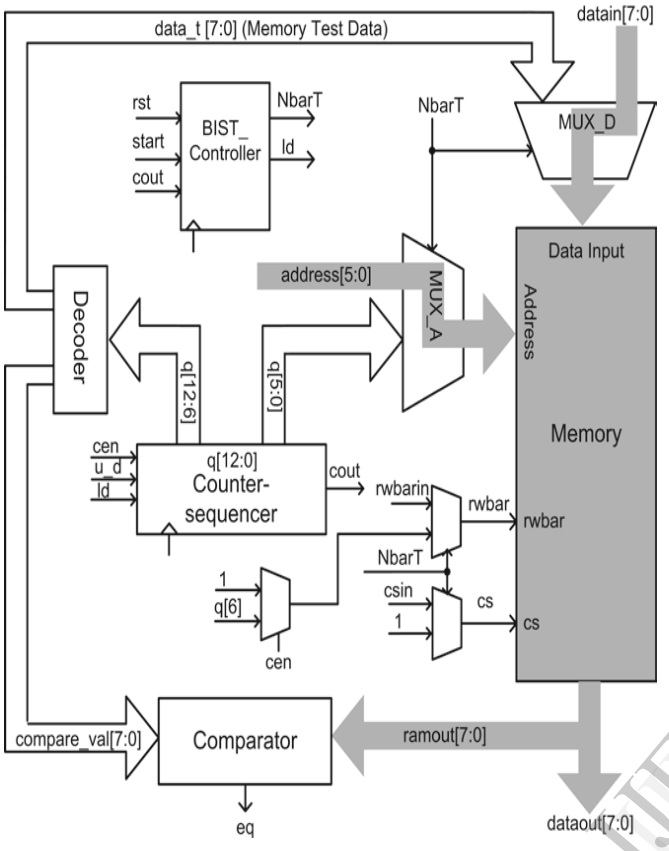

Fig.4. March C- MemoryBIST Architecture

ii) Decoder

The test data decoder uses a 7-bit input vector to generate memory test patterns and the value that is to be compared with the value read from the memory. For every bit of each memory word, all six steps of March C- algorithm (that contains ten operations) are exercised. So using the three most significant input bits, the bit within the word that the above ten operations must be performed on is specified. The decoder specifies which test value must be written into the memory word.

iii) BIST controller

It starts the counter when it receives the start signal. and waits for the carry-out (cout) to become 1.

iv) Multiplexers

The multiplexers of the BIST architecture select between normal memory inputs and BIST provided inputs. When *NbarT* is "0," the memory is working in the normal mode and when this input becomes "1" it operates in test mode.

v) Comparator

Initially, the same test pattern is written into all memory locations, and then these data are read out from all locations. As data are being written and read, the decoder input and thus test patterns remain unchanged. A comparator checks memory data with decoder output. When memory is being tested and data are being read, the comparator should have same data on both its inputs.

IV. Simulation Results

The design entry is modeled using VHDL in Xilinx ISE Design Suite 13.2 and the simulation of the design is performed using ModelSim from Xilinx ISE to validate the functionality of the design. Here design of Bit-Oriented memory test algorithm is simulated.

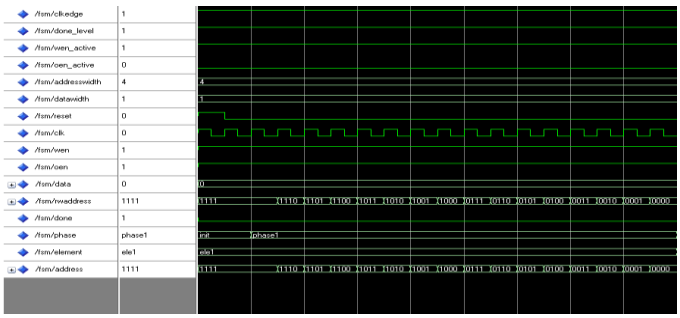

Fig.5.Simulation result of March element M0

March element M0 (phase1) indicates writing data 0 to all memory locations from rwaddress =1111 to rwaddress = 0000 by forcing the signal values ud=0. Here, ele1 indicate this write 0 operation when we=1 and oe=0.

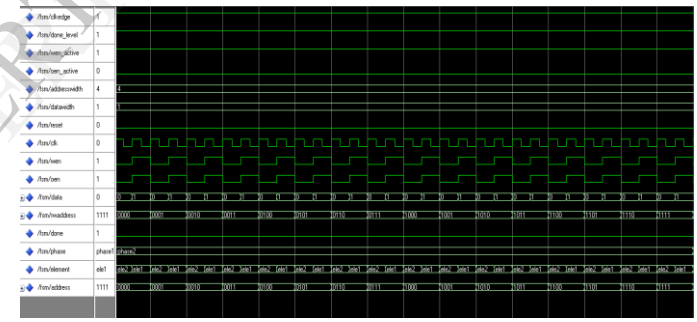

Fig.6.Simulation result of March element M1

March element M1 (phase2) indicates reading data 0 and writing data1 to all memory locations from rwaddress=0000 to rwaddress =1111 by forcing the signal ud=1. Here, ele1 indicates read 0 operation when we=0 and oe=1 and ele2 indicates write 1 operation when we=1 and oe=0.

Similarly, March elements M2, M3, M4, M5 are obtained.

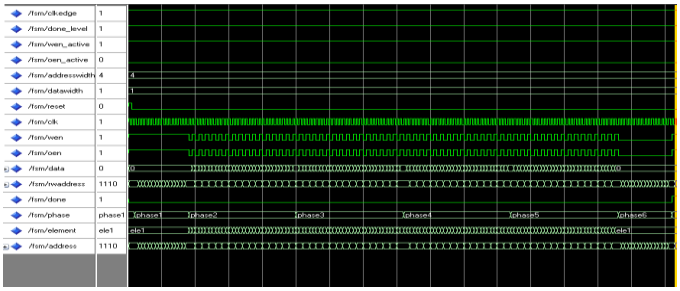

Fig.6.Simulation result of Bit-Oriented March C- algorithm

After the successful completion of all March elements, done signal goes to high .Then it ensures that there is no fault in the memory.

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. A built-in self-test for 8 bit SRAM using March C- algorithm is designed in this project. March C- algorithm uses March C- patterns which identify a huge number of real defects seen in manufacturing test. It can cover host faults including stuck-at faults, transition faults, and many coupling faults and address decoder faults. The project is designed using VHDL and simulated using ModelSim 6.2b design suite from Mentor Graphics.

REFERENCES

[1] M.S.Prasanthi, M.rangaswamy, M.E.dinakar "Restartable BIST controller for fault detection in CLB of FPGA*", International Journal of Emerging Trends in Engineering and Development,* Issue 3, Vol.5 September 2013.

[2] M.H. Husin, S.Y. Leong, M.F.M. Sabri, R. Nordiana "Built in self test forRAM Using VHDL" *2012 IEEE Colloquium on Humanities, Science & Engineering Research (CHUSER 2012),* December 3-4, 2012.

[3] M.I.Masnita, W.H.W.Zuha,R.M.Sidek,an A.H.Izhal, "March-based SRAM diagnostic algorithm for distinguishing Stuck-At and transition faults" ,IEICE Electronics Express,Vol.6,No.15,1091-1097,Aug 2009.

[4] Stroud ,Charles E., " A Designer's Guide To Built- in Self-test",Springer,p.61,2004.

[5] A. van de Goor and I. Tlili, "March Tests for Word-Oriented Memories," Proc. Design Automation and Test in Europe, 1998, pp. 501-508. Feb.2001.

[6] M. L. Bushnell and V. D. Agrawal, " Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits", Boston: Springer, 2000.

[7] Mirron Abramovici, Melvin Breuer and Arthur Friedman, " Digital Systems Testing And Testable Design " ,Piscataway,New Jersy, IEEE press,1994.