PERFORMANCE ANALYSIS OF BIST ALGORITHM FOR RAMs

L. Kusuma
Department of Digital System and Computer Electronics,
Sreenidhi Institute of Science & Technology (Autonomous) Hyderabad, India

Abstract: Design for Testability (DFT) techniques are required in order to improve the quality and reduce the test cost of the digital circuit, while at the same time simplifying the test, debug and diagnose tasks. Built-in self-test (BIST) is a design for testability technique in which a portion of a circuit on a chip, board, or system is used to test the digital logic circuit itself. Traditional march algorithms erase the memory contents prior to the testing. Using the march algoritms with the transparent BIST the content of the memory at the end of the test is identical to that before the test. A march test contains a sequence of march elements which are composed by March tests which can detect several fault models such as Stuck-at Faults (SAF), Address Faults (AF) and some Coupling Faults (CF). A march element is specified by an address order and a finite number of Read/Write operations. Transparent BIST schemes for RAM modules assure the preservation of the memory contents during testing. In transparent BIST the compaction and data generation module is implemented utlizing an ALU. In circuits that contain ALU, the output of the RAM is either directly driven to the inputs of ALU or can be driven using processors instructions. This imposes lower hardware overhead and less complexity in the control circuitry. Transparent BIST can test an array of non-identical memories and therefore the time and the cost required for the test is also reduced.

Keywords- built in self test (BIST), march test, tranaparent BIST

I. INTRODUCTION

A memory unit is a device in which the binary information is transferred for storage and from which information is retrieved when needed for processing. When data is processed, information from memory is transferred to selected registers in the processing unit. The results obtained in the processing unit are transferred back to store in memory. Binary information received from an input device is stored in memory, and information transferred to an output device is taken from memory. A memory unit is a collection of cells that are capable of storing a large quantity of binary information associated with circuits needed to transfer information into and out of a device. The architecture of memory is done in such a way that information can be selectively retrieved from any of its internal locations. Memory tests 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 then verifying this data by reading it back. If all the values read back are the same as those that were written, then the memory device is said to be working properly or fault free, otherwise device fails. Different test methodologies have been evolved from the years to identify the memory defects, one such test is memory built in self test which involves built in self test circuitry for each memory array.

BIST (Built-in Self Test)

Built-in self-test or BIST – installs self-contained test-controllers to automatically test a logic (or memory) structure in the design.

Advantages of implementing BIST:-

1) lower cost of test, since the need for external electrical testing using an ATE will be reduced, if not eliminated
2) better fault coverage, since special test structures can be incorporated onto the chips
3) shorter test times, if the BIST can be designed to test more structures in parallel
4) easier customer support and
5) capability to perform tests outside the production electrical testing environment.

II. PROPOSED ALGORITHMIC METHODS

A. March test algorithms

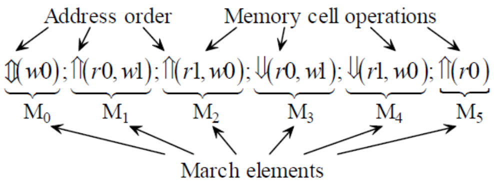
March algorithms are well known for memory testing because March-based tests are simple and possess good fault coverage hence they are the dominant test algorithms implemented in the modern memory BIST. The proposed march algorithm is march c- algorithm which uses march elements for testing the memory. A march test contains a sequence of march elements which is composed by March tests and are able to detect several fault models such as Stuck-at Faults (SAF), Address Faults (AF) and some Coupling Faults (CF). The operations that can be executed in the cells may be-write zero (w0), write one (w1), read zero (r0) and read one (r1). The read operation checks if the value inside the cell is the expected one and the order in which cells are considered can be ascending or descending.

March Test Notation:

A March element is a finite sequence of operations or primitives applied to every memory cell before proceeding to next cell. For example, ↓(r1, w0) is a March element and r0 is a March primitive. The address order in a March element can be increasing (↑), decreasing (↓), or either increasing or decreasing (↕). An operation 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)


Address order
Memory cell operations
$\Updownarrow(w0);\Uparrow(r0,w1);\Uparrow(r1,w0);\Downarrow(r0,w1);\Downarrow(r1,w0);\Uparrow(r0)$
M0
M1
M2
M3
M4
M5
March elements

The advantage of March tests is that, it provides high fault coverage and the test time is usually linear with the size of the memory which makes it acceptable from industrial point of view. March based algorithms were capable of locating and identifying the fault types which can help to detect design and manufacturing errors. But the march algorithms has the disadvantage that it erase the memory contents prior to testing. To prevent the loss of data a transparent approach is introduced.

B. Transparent test

The transparent method avoids traditional comparison and uses a signature analysis mechanism based on a feedback shift register. The concept of transparent BIST where the initial w0 phase is eliminated, and a signature prediction phase is used instead. The signature prediction phase consists of read operations equivalent to all the read operations of the March algorithm and it is utilized in order to calculate a signature that will be compared against the compacted signature calculated during the (remaining) march test. March tests can be easily extended to transparent versions by replacing values 0 and 1, in the read and write operations, by ‘a’ and ‘a^{c}’ respectively, where ‘a’ refers to original content and ‘a^{c}’ to its complement. Besides this modification, the initialization part in the original march test should be removed. Transparent tests use inversion instead of writing predefined values. This allows restoring memory content after test procedure.

The notation for the transparent version of the algorithm differs slightly from the one used in march algorithms. Instead of r0, r1, w0, w1 the notations r_a, r_a^c, w_a, w_a^c and $(r_a)^c$ are utilized where

r_a - Read the contents of a word of the RAM, expecting to read the initial contents of the RAM word (i.e., before the beginning of the test).

r_a^c - Read the contents of a word of the RAM, expecting to read the complement of the initial contents of the RAM word.

$(r_a)^c$ - Read the contents of a word of the RAM expecting to read the initial word contents and feed the complement value to the compactor.

w_a - Write to the memory word; the value that was stored in this memory word at the beginning of the test is (assumed to be) written to the word.

W_a^c - Write to the memory word; the inverse of the value that was stored in this memory word at the beginning of the test is (assumed to be) written to the word.

The general rules for transforming any March test algorithm to a transparent one are presented below:

Step1. Eliminate the memory initialization phase from the test algorithm.

Step 2. If the write operation changes the memory cell state then replace it with writing complemented state (wa^c), in the opposite case
use writing the same state (*wa*).

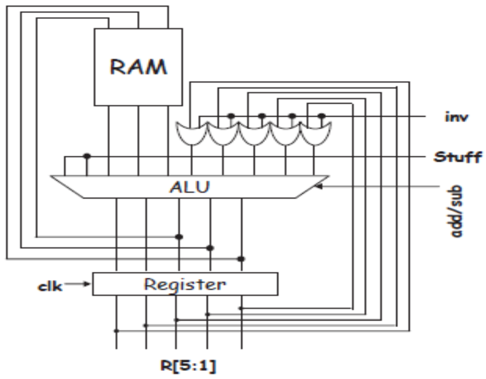
Step 3. Replace each memory read operations *r*0 and *r*1 with *ra* and ra^c operations correspondingly.

Step 4. If the number of performed changes in memory cell contents is odd then add a sequence, which will complement the state of all memory cells. This restores the initial state of the memory.

The obtained test algorithm is called the transparent one.

$$\Uparrow(r_a);\ \Uparrow((r_a)^c);\ \Downarrow(r_a);\ \Downarrow((r_a)^c);\ \Downarrow(_ra);$$
$$\Uparrow(r_a,\ w_a^c);$$
$$\Uparrow(r_a^c,\ w_a);$$
$$\Downarrow(r_a,\ w_a^c);$$
$$\Downarrow(r_a^c,\ w_a);$$
$$\Downarrow(r_a);$$

By default, the data driven to the comparator with the (ra)c operation are identical to the data driven by the rac . The importance of the (ra)c operation is the following: during the signature prediction phase the contents of the RAM are equal to the initial contents (since no write operation has been performed); therefore, in order to simulate the rac operation we invert these contents prior to driving them to the compactor. With the transparent BIST algorithms, the contents of the memory at the end of the test are identical to those before the test. Also, since the read elements of the signature prediction phase (Mo) are identical to the read elements of the testing phase (M1 – M5), then if we store the result of the compaction of M0 and compare it to the result of the compaction of (M1 – M5), then we can detect faults that occur due to the write operations of the march algorithm. In transparent BIST, signature prediction phase is used and write all zero phase is neglected, during which signature is captured and is stored. The final signature is compared against the captured signature to check whether the fault is present in the RAM word or not. The need to capture the contents of the data in memory at the beginning of transparent BIST test imposes the need to employ Multiple Input Shift Registers (MISR) structures which increasing the hardware overhead and also the memories with same word can only be tested.



Architecture implementing transparent test using 5-bit ALU.

Transparent BIST has the disadvantage that the signature prediction phase adds up to the total testing time with a percentage of 30%. In order to overcome this problem, the concept of symmetric transparent BIST can be used. This scheme cannot be apply to test memories having different word width. Hence this requires separate BIST modules and increases hardware overhead.

C. Symmetric transparent test

A symmetric transparent test poses a constraint on the symmetry of the march test, i.e it should have the same number of reading for the original and the complement content. The signature mechanism computes the signature when fed by the original content and computes the reciprocal signature when fed by the complementary content. By doing this, the initial state of the signature mechanism should be found at the end of the test when the memory is fault free.

The notations used in symmetric transparent online BIST are:

r_a - Read the contents of a word of the RAM, expecting to read the initial contents of the RAM word (i.e., before the beginning of the test).

r_a^c - Read the contents of a word of the RAM, expecting to read the complement of the initial contents of the RAM word.

$(r_a)^c$ - Read the contents of a word of the RAM expecting to read the initial word contents and feed the complement value to the compactor.

w_a - Write to the memory word; the value that was stored in this memory word at the beginning of the test is (assumed to be) written to the word.

W_a^c - Write to the memory word; the inverse of the value that was stored in this memory word at the beginning of the test is (assumed to be) written to the word.

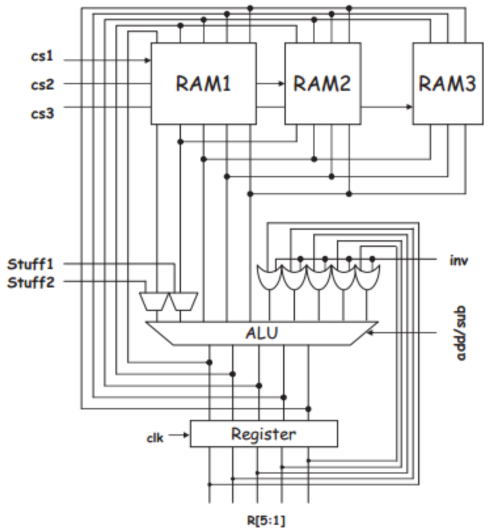
$$\Uparrow(ra^c);\ \Uparrow(ra, wa^c);\ \Uparrow(ra^c, wa);\ \Downarrow(ra, wa^c);\ \Downarrow(ra^c, wa);\ \Downarrow(ra)$$

In Symmetric Transparent BIST, the signature prediction phase is skipped and the march series is modified in such a way that the final signature is equal to all zero state, irrespective of the initial contents of RAM. In Symmetric Transparent BIST, data generation module is implemented by using an ALU. The output of RAM is given directly to the inputs of ALU or by using processor instructions. This scheme imposes lower hardware overhead and less complexity than previously proposed scheme. This scheme uses an ALU to generate test patterns and the word width of memory can be smaller or can be equal to the number of stages in ALU. Hence multiple non-identical memories can be tested in a pipeline fashion and the area cost is reduced.

III. IMPLEMENTED ALGORITHM

The architecture of the Symmetric Transparent Online Test for word oriented RAMs consists of different modules such as:

1. ALU
2. Register
3. OR gates
4. Three RAM modules



A. ALU Module

The ALU module in the architecture stands for Arithmetic and Logic Unit which performs arithmetic and logical operations on the data. If there is a carry bit during addition then add '1' from result, if there is a borrow bit during the subtraction then subtract '1' from the result.

B. RAM module

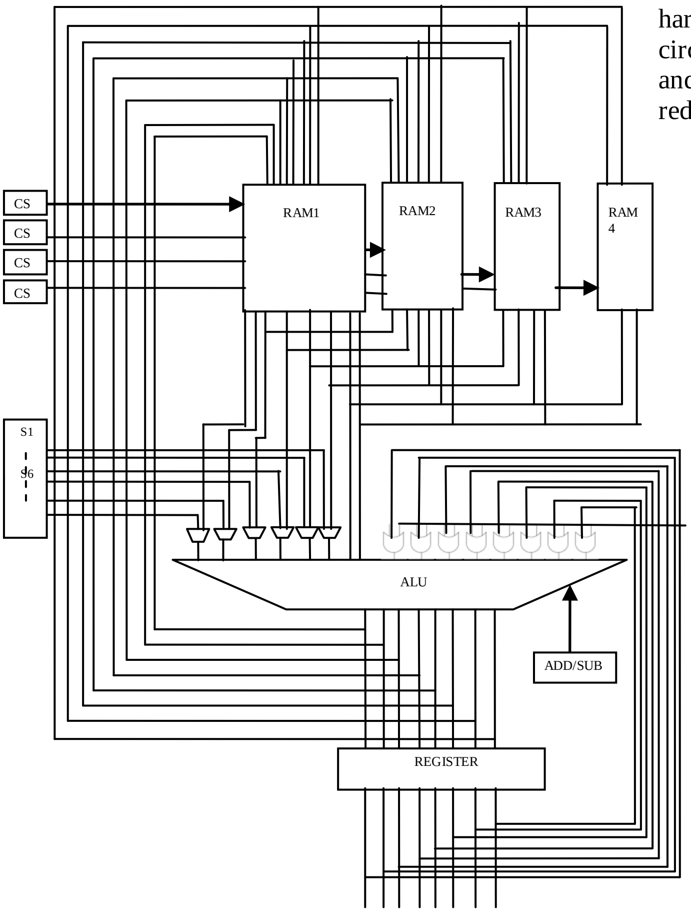
The RAM module implemented consists of two modes of operations such as Read & Write. The RAM module during Write operation address is given and the data that has to be stored is also given. The data will be stored in the specified address. During Read operation the address is specified. The data that is present in the specified address is given on the output data signal. In the architecture three RAMs are implemented with different word widths. The RAMs can store the 3bits, 4bits and 5bits respectively. The RAMs are selected depending upon the select signals cs1, cs2 and cs3.

C. Stuff

Stuff signals are connected to the output signals of the RAMs that are passing through the ALU. If RAM of 3bits is being accessed then the Stuff values must be given in order to make it 5 bits. If RAM of 5 bits is being accessed then the Stuff values must not be given because it is already 5bits.

Working RAM1 is of 3bits, RAM2 is of 4bits, RAM3 is of 5bits. These are selected with the help of Chip select (CS). When CS1 is enabled, RAM1 is selected for testing. When CS2 is enabled, RAM2 is selected for testing. When CS3 is enabled, RAM3 is selected for testing. When RAM1 is selected then the stuff values are given as "00". Because the ALU that we are implementing is of 5-stage. When RAM2 is selected then the stuff values were '0'. There is no stuff value when we have selected RAM3. Let us consider the 3-bit RAM1 presented in Fig. The outputs of the memory are driven to an n = 5-stage ALU comprising a 1's complement adder. For the implementation of the $(ra)^c$ march element, the subtraction operation of the accumulator can be utilized. In order to apply march elements of the form (r_a, w_a^c) or $(r_a^c,$

w_a) the output of the RAM must be inverted and then fed back to its inputs; with the proposed scheme, this can be done by forcing the all-1 vector to one input of the adder/subtractor and perform a subtract operation. This is done with the OR-gates whose one input is driven by the inv signal. Therefore, the inverse of the read vector appears at the outputs the adder/subtractor and applied to the RAM inputs.
The similar architecture of symmetric transparent algorithm can be implemented for ram of larger bits. Depending on number of bits of ram the ALU module also changes. The bits of ram and the ALU module should be equal to perform the test. Here we implement this test on 8-bit RAM, so the ALU module is also designed for 8-bits.



Proposed architecture using 8-bits

But testing only a single RAM on a single ALU module utilizes more hardware for a particular RAM. As this ALU is designed for 8-bits, it can be extented to test more RAM's which are lesser than 8-bits. So the RAM's of 6-bit, 4-bit and 2-bit are tested on same ALU with the help of stuff bits. As the ALU is of 8-bit, the input and the output of the ALU is of 8-bits. Here we use stuff bits to balance the data bits. Least number of bits tested are 2 so it require 6-stuff bits to make the data bits of 8. For 4-bit RAM, 4 stuff bits are required and for 6-bits RAM 2stuff bits. The symmetric transparent algorithm is used for testing these four RAM's which ensures the preservation of data after testing and reduces the time by testing the words of different word width on same ALU and also cost.

IV. CONCLUSION

To overcome the dis-advantages of march c- algorithm and the transparent test, symmetric transparent test is proposed which is implemented using ALU which ensures the testing of RAM without erase of memory. RAMs of different word width can be tested on the same ALU module which reduces the testing time and the cost. By utilizing an ALU module RAM can be tested where the number of stages of ALU can be larger than the RAM word, where the largest RAM width does not exceed the number of stages of ALU. This also reduces the hardware overhead. This imposes lower hardware overhead and less complexity in the control circuitry. An array of non-identical memories can be tested and therefore the time and the cost required for the test is reduced and multiple faults can be detected.

REFERENCES

[1] Karl Thaller and Andreas Steininger "A Transparent Online Memory Test for Simultaneous Detection of Functional Faults and Soft Errors in Memories" IEEE TRANSACTIONS ON RELIABILITY, VOL. 52, NO. 4, DECEMBER 2003

[2] Ioannis Voyiatzis, "Accumulator – based compression in Symmetric Transparent RAM BIST"

[3] Nicolaidis M., "Transparent BIST for RAMs", International Test Conference, 1992, pp. 598-607

[4] Voyiatzis I., "Test Vector Embedding into Accumulator-generated sequences: A Linear-Time Solution", IEEE Transactions on Computers, April 2005.

[5] Ad. J. Van de Goor, Issam B.S. Tlili, "A Systematic method for modifying March tests for bit-oriented Memories into tests for Word-Oriented Memories", IEEE Trans. on Computers, vol. 52, no. 10, October 2003, pp.1320-1330.

[6] Robert C. Aitken, "A Modular Wrapper Enabling High Speed BlST and Repair for Srnall Wide Memories" ITC INTERNATIONAL TEST CONFERENCE,2004.

[7] D. C. Huang and W. B. Jone, "A Parallel Transparent BIST Method for Embedded Memory Arrays by Tolerating Redundant Operations" IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 21, NO. 5, MAY 2002.

[8] Xiaogang Du1, Nilanjan Mukherjee1, Chris Hill1, Wu-Tung Cheng1, Sudhakar Reddy "A Field Programmable Memory BIST Architecture Supporting Algorithms with Multiple Nested Loops"

[9] Ioannis Voyiatzis, "An ALU-Based BIST Scheme for Word-Organized RAMs" IEEE TRANSACTIONS ON COMPUTERS, VOL. 57, NO. 5, MAY 2008.

[10] Jin-Fu Li, “Transparent-Test Methodologies for Random Access Memories Without/With ECC” IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 26, NO. 10, OCTOBER 2007

[11] Michael Nicolaidis, “Theory of Transparent BIST for RAMs” IEEE TRANSACTIONS ON COMPUTERS, VOL. 45, NO. 10, OCTOBER 1996

[12] Xiaogang Du, Nilanjan Mukherjee, Wu-Tung Cheng, “Full-Speed Field-Programmable Memory BIST Architecture” INTERNATIONAL TEST CONFERENCE 2005 IEEE