IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 4, Issue 04, November 2015)

IJDACR ISSN: 2319-4863

Efficient BIST architecture for combinational and sequential faulty circuits

Ankita Tripathi A.P RAO, (Reader) (Et &T dept.),
ankita9tripathi@gmail.com prabhakar.amarana@gmail.com

Abstract— **Very Large Scale Integration (VLSI) has made an extraordinary effect on the development of integrated circuit technology. It has not only decreased the dimension and the price but also improved the complexness of the circuits. There are, however, prospective issues which may slow down the efficient use and development of upcoming VLSI technology. Among these is the issue of circuit testing, which becomes progressively challenging as the range of integration grows. Because of the high device counts and restricted input/output accessibility that define VLSI circuit, conventional testing techniques are often worthless and inadequate for VLSI circuit. Built-in self-test (BIST) is a commonly used design technique that allows a circuit to test itself. In this paper BIST architecture is implemented for testing of various faulty circuits. Also testing for embedded memory, MARCH-Y algorithm is used for coupling fault, stuck at on fault, stuck at zero faults.**

Keywords— **VLSI, BIST, Faulty Circuits, MARCH-Y**

I. INTRODUCTION

BIST (Built-In Self-Test) for random logic is becoming an eye-catching substitute in IC testing, although logic BIST is a recent subject which is under research over more than 3 decades. This paper provides the use of a deterministic logic BIST structure upon state-of-the-art industrial circuits. Nevertheless, new innovations throughout deep-submicron IC process engineering as well as core-based IC design and design engineering will surely lead to more popular using logical BIST due to the fact outer assessment is actually becoming a lot more difficult as well as high-priced. Logic built-in self-test (BIST) depend on the fundamental design for test methodology. For any testing methodology, the following factors should be considered- high and easily verifiable fault coverage, minimum test pattern generation, minimum performance degradation, at-speed testing, short testing time, and reasonable hardware overhead [1]. Logic Built-In Self-Test (BIST) provides a feasible solution to the above demands. First, BIST significantly reduces off-chip communication to overcome the bottleneck caused by the limited input/output access. Further, it eliminates much of the test pattern generation and simulation process [1]. Testing time can be shortened by testing multiple units simultaneously through test scheduling. Hardware overhead can be minimized by careful design and through the sharing of test hardware. In the modern System-on-aChip (SoC) design, many cores are integrated into a single chip. Some of them are embedded, and cannot be accessed directly from the outside of the chip. Such SoC designs make the test of these embedded cores a great challenge [2]. BIST is one of the most popular test solutions to test the embedded cores. Since more and more transistors are integrated on a single IC, the amount of test vectors to test such large ICs is increasing. This requires large memories in external test equipment. In addition, a significant increase is predicted.

Originally, the predominant compelling purpose for the adopting of BIST was the need to execute in-field examining. Just lately, there have been developing desires for BIST as it may lower the price of manufacturing test together with strengthen the standard of the particular test by providing at-speed testing ability. In BIST, pseudorandom styles tend to be generated on chip; the actual replies tend to be compacted about chip, as well as the handle impulses tend to be pushed simply by an on-chip controller. The amount of examination files exchanged with the tester is consequently considerably lowered. In addition, the scan cells are configured into a large number of relatively short scan chains, thus reducing the time required to apply a single test pattern. The low memory and performance requirements on the tester allow the usage of very low cost testers for manufacturing test of designs with logic BIST.

II. METHODOLOGY

Linear Feedback Shift Register

LFSR is an n-bit shift register which pseudorandomly scrolls between 2n-1 values, but does it very quickly because there is

IJDACR
ISSN: 2319-4863

International Digital & Journal of Application

IJDACR
International Journal Of Digital Application & Contemporary Research

Contemporary research
Website: www.ijdacr.com (Volume 1, Issue 1, August 2012)

minimal combinational logic involve [1]. The all zeros case is not possible in this type of LFSR, but the probability of any bit being "1" or "0" is 50% except for that. Therefore, the sequence is pseudorandom in the sense that the probability of a "1" or "0" is approximately 50%, but the sequence is repeatable. Like a binary counter, all 2^n - 1 states are generated, but in a "random" order that is repeatable. The exclusive-OR gates and shift register act to produce a pseudorandom binary sequence (PRBS) at each of the flip-flop outputs. By correctly choosing the points at which we take the feedback from an n -bit shift register we can produce a PRBS of length $2^n - 1$, a maximal-length sequence that includes all possible patterns (or vectors) of n bits, excluding the allzeros pattern. In an LFSR, the bits contained in selected positions in the shift register are combined in some sort of function and the result is fed back into the register's input bit. Fig.1 shows a 3bit LFSR.

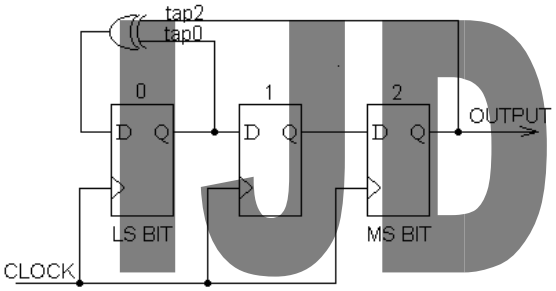


Figure1: 3 bit maximal-length LFSR

The feedback is done so as to make the system more stable and free from errors. Specific taps are taken from the tapping points and then by using the XOR operation on them they are feedback into the registers.

Signature Analysis

Signature Analysis is a compression technique based on the concept of cyclic redundancy checking [2]. The good and faulty circuits produce different signatures. Test Patterns for BIST can be generated at-speed by an LFSR with only a clock input. The outputs of the circuitunder-test must be compared to the known good response. In general, collecting each output response and off-loading it from the circuit under test for comparison is too inefficient to be practical. The general solution is to compress the entire output stream into a single signature value.

SISR - Single-Input Signature Register

A serial-input signature register can only be used to test logic with a single output. There are several ways to connect the inputs of LFSRs to form an SISR. Since the XOR operation is linear and associative, $(A \oplus B) \oplus C = A \oplus (B \oplus C)$, as long as the result of the additions are the same then the different representations are equivalent. If we have an n -bit long SISR we can accommodate up to n inputs to form the signature. If we use $m < n$ inputs we do not need the extra XOR gates in the last $n - m$ positions of the SISR. SISR reduce the amount of hardware required to compress a multiple bit stream. LFSR and/or SISR circuit is implemented using a memory already existing in a circuit to be tested. If we apply a binary input sequence to LFSR, the shift register will perform data compaction (or compression) on the input sequence. At the end of the input sequence the shift-register contents, Q_0, Q_1, and Q_2, will form a pattern that we call a signature. If the input sequence and the serial-input signature register (SISR) are long enough, it is unlikely (though possible) that two different input sequences will produce the same signature. If the input sequence comes from logic that we wish to test, a fault in the logic will cause the input sequence to change. This causes the signature to change from a known good value and we shall then know that the circuit under test is bad. This technique, called signature analysis, was developed by Hewlett-Packard to test equipment in the field in the late 1970s. The simplest form of this technique is based on a single input LFSR.

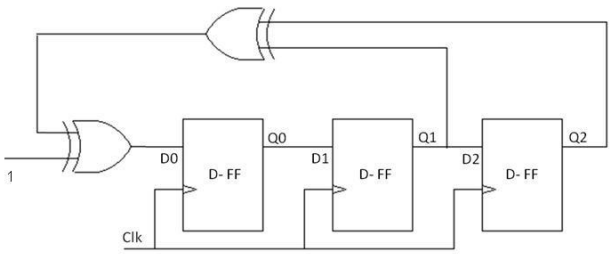


Figure2: 3 bit Single-input signature register (SISR)

Every LFSR has a characteristic polynomial that describes its behavior. Degree of polynomial is given by the number of shift registers.

Description of MARCH Y algorithm

March Y algorithm: ↕ (W0); ↑ (R0, W1, R1); ↓ (R1, W0, R0); ↕ (R0)

The basic notations used in algorithm are as follows:

- ↑: address 0 to n-1
- ↓: address n-1 to 0

International Digital & Journal of Application

IJDACR
International Journal Of Digital Application & Contemporary Research

Contemporary research
Website: www.ijdacr.com (Volume 1, Issue 1, August 2012)

- ↕: either way
- W0: Write 0 to the word
- W1: Write 1 to the word
- R0: Read a cell whose value should be 0
- R1: Read a cell whose value should be 1
- The FSM should have a START state and be activated by an external signal RUN_BIST.
- The design should have an input signal BIST_COMP feedback from the SRAM such that when BIST_COMP=1 means there is a fault, 0 no fault. If BIST-COMP=1, the simulation should be terminated and the FSM goes to END state. The address lines should read the failing address.
- Output pins Read_En to be activated during the Read operation, and Write_En to be activated during the write operation.
- A Compare state follows each Read operation, during which NO Read or Write operations are activated
- An output signal called Error is activated when the FSM goes to END state corresponding to failure.
- The machine should produce the address and data output signals to feed the array.
- You should provide a test fixture file to run the machine.
- Use 500MHZ clock.

Design Methodology

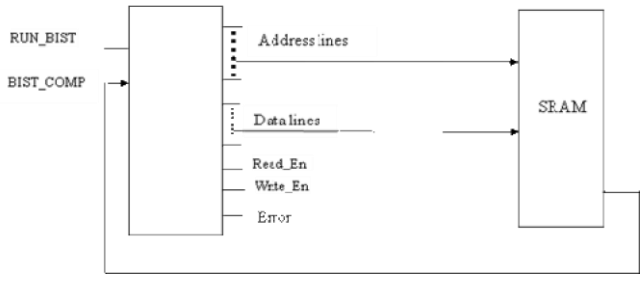


Figure3: BIST machine Specifications

The BIST March Y algorithm is implemented using a Finite State Machine. The state diagram used to design the BIST is shown below:

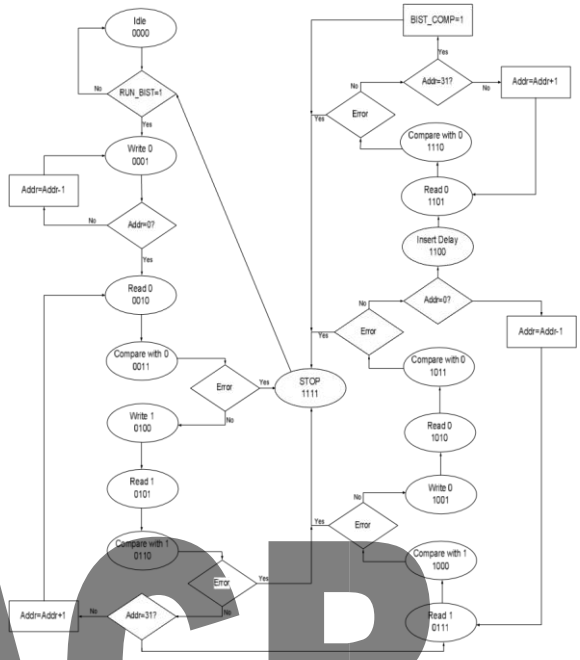


Figure4: FSM based implementation of the MARCH Y BIST algorithm

The finite state machine of the BIST March Y algorithm has 16 states, hence the state are represented using 4 bits. The controller starts the BIST algorithm if the RUN_BIST signal is asserted, and otherwise it stays in the idle state. If the RUN_BIST=1, then initially write-0 operation is performed on all the words sequentially and then a read-0 is performed and compared with 0. If the result is correct then we write-1 on the same cell. If the comparison is incorrect then the machine goes to STOP state. Again, a read-1 operation is performed on the same cell and compared with-1. If the comparison result is correct, it increases the address and if the comparison is incorrect, the machine goes to STOP state.

Similar to the above process, initially read operation takes place; this read is followed by a comparison operation. And then write-0 is performed and again read and comparing with 0 takes place.

In the end, read 0 operation is performed and compared with 0. In all the above comparisons, if the result s incorrect then the machine goes to idle state otherwise the machine goes to the corresponding state.

The state transitions are shown in the above figure. After each comparison the machine goes to either write state or to the STOP state if the FSM generates an error. The FSM is easily designed using case and if statements in the VHDL. In each state the corresponding Read enable (RE) and Write enable (WE) signals are asserted.

IJDACR
ISSN: 2319-4863

International Digital & Journal of Application

IJDACR
International Journal Of Digital Application & Contemporary Research

Contemporary research
Website: www.ijdacr.com (Volume 1, Issue 1, August 2012)

If the machine is in STOP state the Finish is always asserted and if the error is found, the Error signal is also asserted.

The VHDL Implementation of BIST March Y Algorithm

The BIST is hierarchically implemented in VHDL, with the top module as BIST and then BIST controller and SRAM modules are the sub modules in the top BIST module.

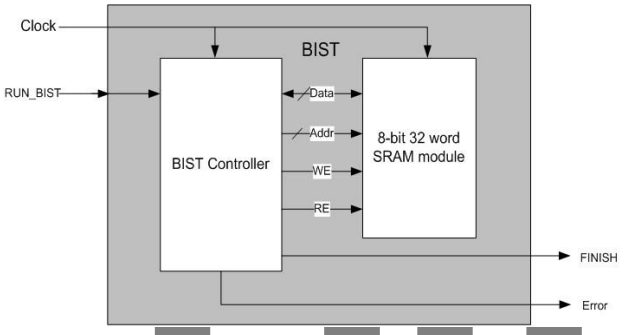


Figure5: Module hierarchy in the VHDL implementation of BIST March Y algorithm

Since this is simulation, error in SRAM will not appear as unless intentionally introduced, we will have one SRAM module which has random bit sequences at some registers which will work as an error in SRAM. Since in the simulation using Modelsim, the error cannot be introduced during simulation hence a separate SRAM module is designed which has an error. According to this the top module is modified to invoke the SRAM file with error and without error. *Faulty Circuits*

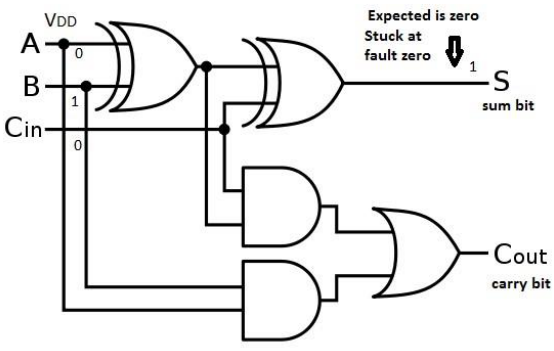


Figure6: Full Adder

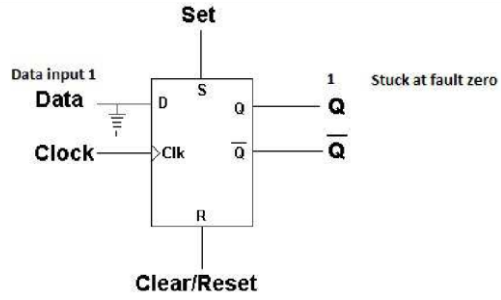


Figure7: D Flip-Flop

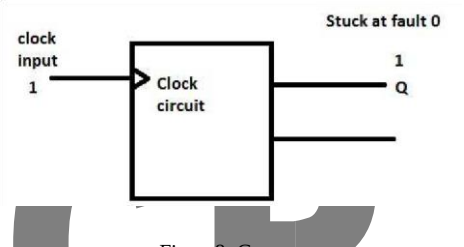


Figure8: Counter

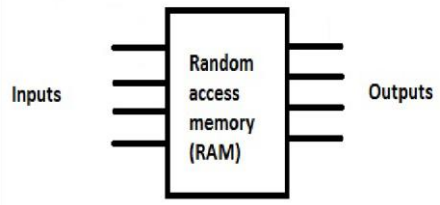


Figure9: RAM

III. SIMULATION RESULTS

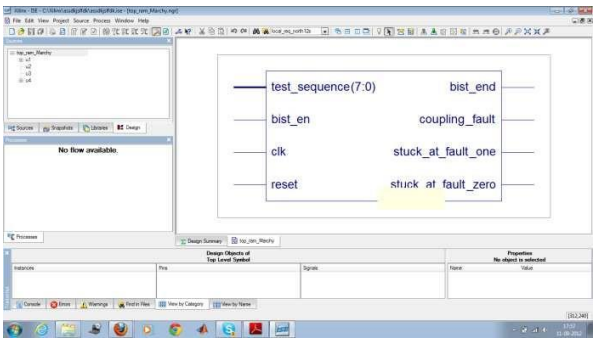
Following figures shows the synthesis result:

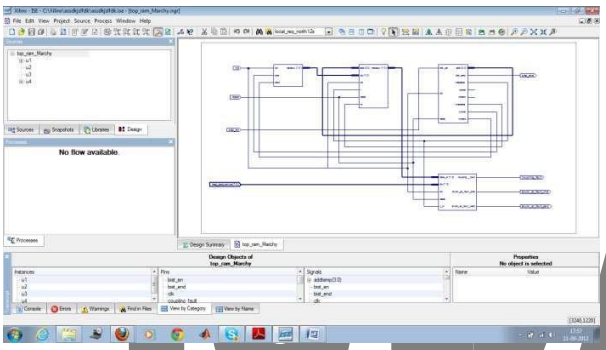
IJDACR
ISSN: 2319-4863

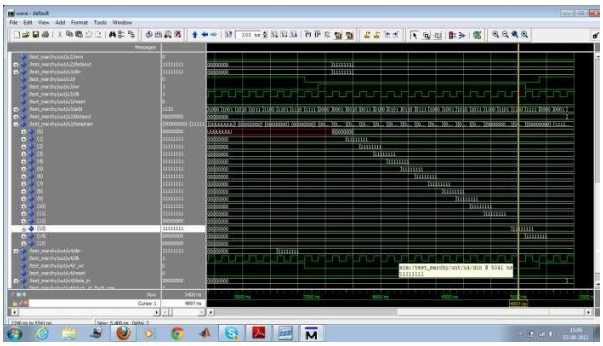
International Digital & Journal of Application

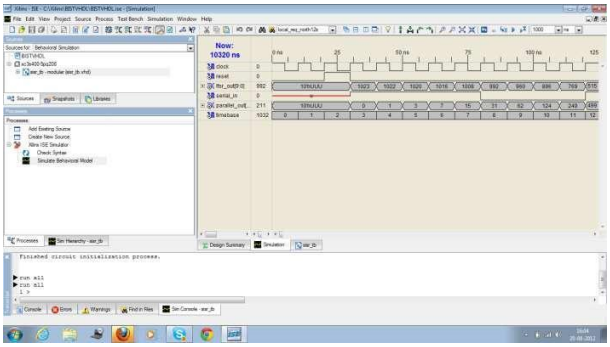
IJDACR
International Journal Of Digital Application & Contemporary Research

Contemporary research
Website: www.ijdacr.com (Volume 1, Issue 1, August 2012)









IV. CONCLUSION

The BIST architecture proposed is implemented using VHDL language and tested on various faulty circuits. Then design has been synthesised on Xilinx and fault has been created and simulated on Modelsim. Special testing for embedded memory MARCH-Y algorithm has been tested for coupling fault, stuck at on fault, stuck at zero faults.

REFERENCES

[1] Digital system testing & testability. Abromovici.

[2] L.T. Wang, Cheng- Wen Wu and Xiaoqing Wen, "VLSI Test Principles & Architectures Design for testability".

[3] Wu- Tung Cheng, Manish Sharma, Thomas Rinderknecht, Liyang Lai and Chrisshill, "Signature based diagnosis for logic BIST", ITC, 2006.

[4] N. Tamarapalli and J. Rajski, "Constructive Multi- Phase Test Point Insertion for Scan-Based BIST", Proc. of International Test Conference, pp. 649-658, 1996.

[5] A. Hassan, J. Rajski, R. Thompson and N. Tamarapalli, "Method and Apparatus for At-Speed Testing of Digital Circuits", US patent pending.

[6] B. Nadeau-Dostie, D. Burek and A. Hassan, "Scan-BIST: A Multifrequency Scan-Based BIST Method", IEEE Design &Test of Computers, pp. 7-17, Vol. 11, No. 1, Spring 1994.