Design of Improved Built-In-Self-Test Algorithm (8n) for Single Port Memory

Manoj Vishnoi, Arun Kumar, Minakshi Sanadhya

***Abstract* — This paper presents a modified March (8n) test algorithm to the Built-In Self-Test (BIST) for Single Port Memory. In this algorithm, test patterns are complemented to generate state-transitions that are needed for the detection of frequently occurring as well as newer occurring faults with the shrinking of channel length. The test pattern will be generated for the single port memory. The use of 8n pattern to generate state transition allow to reducing both of time and energy for detection of faults. As a result, the number of test patterns required is very less than of the traditional method, while the extra hardware is negligible.**

***Keywords*—March Algorithm, BIST (Built-In-Self-Test), Channel Length, Faults, Test-Pattern.**

I. INTRODUCTION

Very Large Scale Integration (VLSI) has had a dramatic impact on the growth of digital technology. VLSI has not only reduced the size and the cost but also increased the complexity of the circuits. This has brought significant improvements in performance. These welcomed improvements have resulted in significant performance/cost advantages in VLSI-implemented systems. There are, however, potential problems which may retard the effective use and growth of future VLSI technology. Among these is the problem of circuit testing, which becomes increasingly difficult as the scale of integration grows. Semiconductor memories are widely used in modern electronics devices. Memory is a regular/symmetrical structure which makes them the most densely packed devices among all types of integrated circuits (IC). Developments in VLSI technology result in a continuously increasing density of memory chips, and the number of components per chip has quadrupled every four years. The exponential increase in density creates great challenge for memory testing. As the feature size of components shrinks, the sensitivity to faults also increases while the faults become more complex. Furthermore, test time grows at least linearly as the number of storage elements per chip increases. However, test cost can not grow at such a pace since the price per storage element drops dramatically as the density increases. Recent development in system-on chip technology makes it possible to incorporate large embedded memory into a chip; whoever, it also complicate the test process as usually there is no direct control to the embedded memory from the outside environment. Built-In Self-Test (BIST) can solve the above memory testing problems which increases the predictability (Controllability + Observability).

Manuscript Received on November 2012

Manoj Vishnoi, Department of ECE, SRM University NCR Campus modinagar India.
Arun Kumar, Department of ECE, SRM University NCR Campus modinagar India.
Minakshi Sanadhya, Department of ECE, SRM University NCR Campus modinagar India.

Test patterns generated by a BIST controller can be either deterministic *or* pseudorandom. Deterministic algorithms (e.g., March algorithms) can achieve higher fault coverage with fewer numbers of test patterns; however, the circuit implementation is more complicated. On the other hand, modified March (8n) testing requires less test pattern while the controller circuits are much simpler. Moreover, for some complicated fault models, march 8n testing is the only feasible solution. In this project we present a new testing algorithm based on March patterns. This method greatly reduces required number of test patterns while the silicon area overhead is minimal.

A. Built in Self Test Architecture

The basic BIST architecture is composed of three hardware modules in addition to the circuit under test (CUT). The architecture is shown in. The functions of these blocks are as follows. The test pattern generator generates the test patterns for the CUT. The response analyzer compresses and analyzes the test responses to determine correctness of the CUT. The BIST controller is the central unit to control all the BIST operations.

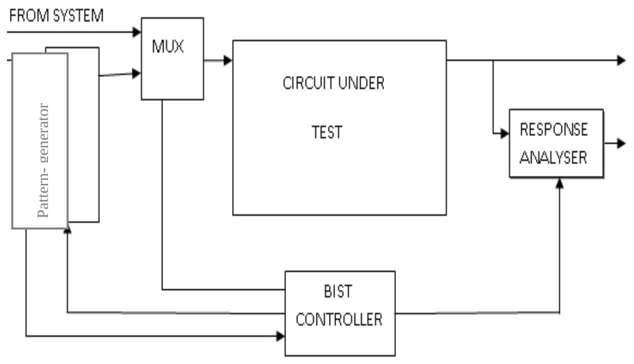

Fig 1: Memory with BIST circuitry

In a BIST system hierarchy, there are BIST controllers at each level of the circuit hierarchy, such as module, chip, board, and system levels. Each BIST controller is responsible for the self test in that particular level, the controls of BIST operations for the lower level BIST, and the report of the test results to the upper level. The design of a test generator is determined by the test strategy being deployed. The test strategy being selected is determined by the fault coverage, test hardware overhead, and testing time. The commonly seen test strategies include the followings

II. MEMORY TESTING AND FAULT TYPES

Memories fail in a number of different ways. The three main parts, address decode logic, memory cell array, and read/write logic,

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

can each have flaws that cause the device to fail. Memory testing, while similar to random logic testing, focuses on testing for these memory-specific failures. The basic types of memory faults include stuck-at, transition, coupling, and Neighborhood pattern sensitive.

Table 1: Functional model of a RAM

Subset of functional memory faults			
	Functional Fault		**Functional Fault**
A	Cell Stuck	i	Address line Stuck
B	Driver Stuck	j	Open in address line
C	Read/Write line Stuck	k	Shorts between address lines
D	Chip- select line stuck	l	Open decoder wrong address
E	Data line stuck	m	wrong address
F	Open in data line	n	Multiple Access
G	Short between data lines	o	Cell can be only set to either 0 or 1
H	Crosstalk between data lines	p	Pattern sensitive interaction between cells

Table 1 shows the functional model of a dynamic random access memory (DRAM). In this model, the internals of the memory are partly visible; hence it is also referred to as the gray box model. This model can also be reused for modeling faults in synchronous RAM (SRAM), read only memory (ROM) or electrically programmable ROM (EPROM). This can be achieved by adjusting some of the blocks shown in the figure. For example, to model SRAM, one needs to discard the refresh logic block. One of the main advantages of functional models is that they have enough details of data paths and adjacent wires.

A. Stuck-at Faults

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

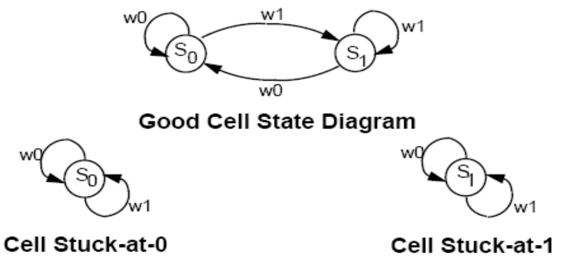

Fig 2: Stuck at fault state diagram

B. Transition Faults

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

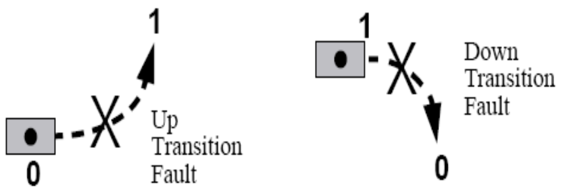

Fig.3: Transition fault

C. Coupling Faults

A coupling fault (CF) between two cells causes a transition in one cell to force the content of another cell to change. The 2-coupling fault model [8], which involves only two cells, is defined as follows: a write operation that generates a ↑ or ↓ transition in one cell changes the content of the second cell. The 2-coupling fault is a special case of the k-coupling fault [8]. A k-coupling fault uses the same two cells as the 2-coupling fault, however it allows the fault to occur only when another k − 2 cells are in a certain state.

• The inversion coupling fault (CFin) is a special case of the 2-coupling fault. It means that a ↑ or ↓ transition in one cell inverts the content of the second cell [8].

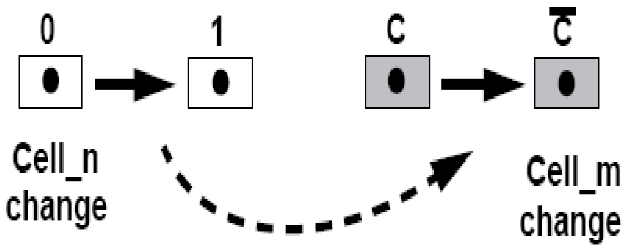

Fig 4: Inversion coupling Fault *Stuck-at Faults*

A memory fails if one of its control signals or memory cells remains stuck at a particular value. Stuck-at faults model this behavior, where a signal or cell appears to be tied to power (SA1) or ground (SA0).

The idempotent coupling fault (CFid) is another particular case of the 2-coupling fault. It means that a ↑ or ↓ transition in one cell forces a second cell to a certain value, 0 or 1.

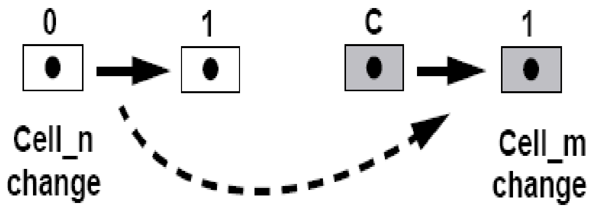

Fig 5: Idempotent Coupling Fault

- The dynamic coupling fault (CFdyn) is a more general case of the CFid. According to its definition a read or write operation on one cell forces the contents of the second cell either to 0 or 1 [3].
- The bridging fault (BF) is caused by a short circuit between two or more cells or lines. It is determined by a logic level rather than a transition write operation. There are two kinds of bridging faults: AND bridging fault (ABF), in which the logic value of the bridge is the AND of the shorted cells or lines, and OR bridging fault (OBF), in which the logic value of the bridge is the OR of the shorted cells/lines.

- In the state coupling fault (SCF) a coupled cell or line is forced to a certain value (0 or 1) only if the coupling cell is in a given state. It is also determined by a logic level.

D. Retention Faults (RF)

A cell fails to retain its logic value after some time. This fault is caused by a broken pull-up resistor [12].

E. Neighborhood Pattern Sensitive Faults

A pattern sensitive fault (PSF) causes the content of a cell (or the ability to change the content) to be influenced by the contents of other memory cells, which may be either a pattern of 0s and 1s or transitions in memory contents.[8].

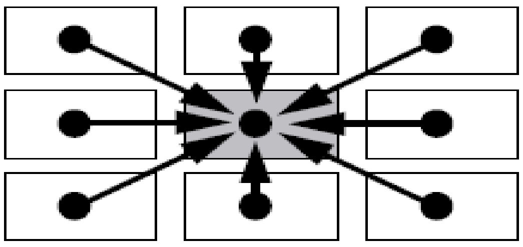
Fig 6: Neighborhood Pattern Sensitive Fault

F. Address Decoder Faults

Address decoder faults (AFs) represent faults in the combinational logic of the address decoder. Two assumptions are generally accepted: the faults do not introduce sequential behavior in the address decoder and the faults will manifest identically during read and write operations. To simplify the problem, we first consider bit-oriented memories, in which only one bit data is stored in each memory location. The functional faults within the address decoder can be classified into four AFs [9].

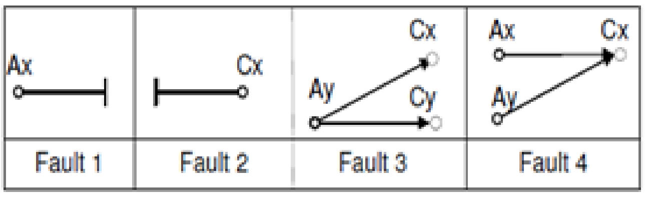

Fig 7: Address Decoder Fault

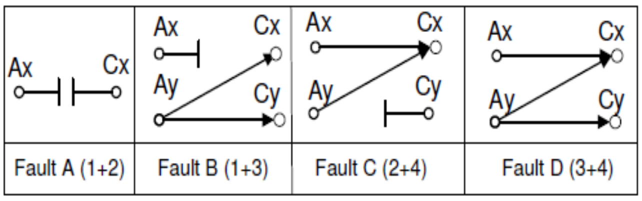

Fig8: Combinations of Address Decoder Faults

- Fault 1: For a certain address, no cell will be accessed.
- Fault 2: A certain cell can never be accessed by any address.
- Fault 3: For a certain address, multiple cells are accessed simultaneously.
- Fault 4: A certain cell can be accessed by multiple addresses.

For bit-oriented memories, because each cell is linked to a dedicated address, none of the faults listed above can stand alone. For example, when fault 1 occurs, then either fault 2 or fault 3 will occur as well.

G. Linked Faults

LFs describe an interesting type of faulty behavior that takes place when more than one FP is sensitized in a defective memory. The definition of an LF is as follows:

LF1 = FP1 $\longrightarrow$ FP2

This means that linked fault 1 consists of FP1 linked to FP2 (i.e., FP1and FP2 are the linking FPs).

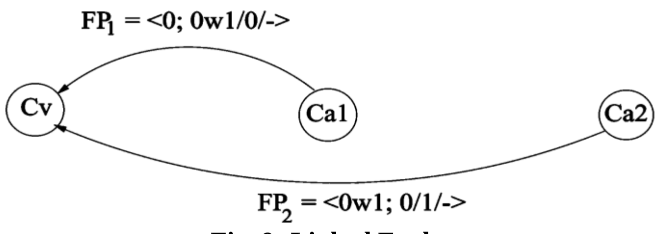

Fig. 9: Linked Faults

III. ALGORITHM

Based on the used memory fault models, memory test algorithms can be divided into four categories [8] as described below:

1. Traditional tests including Zero-One, Check board, GALPAT and Walking 1/0, sliding diagonally and Butterfly. They are not based on any particular functional fault models and over time have been replaced by improved test algorithms, which result in higher fault coverage and equal or shorter test time.
2. Tests for stuck-at, transition, and coupling faults that are based on the reduced functional fault model and are called March test algorithms
3. Tests for neighborhood pattern sensitive faults.
4. Other memory tests: any tests which are not based on the functional fault model are grouped in this category.

As mentioned in Section 2.1, March test algorithms can efficiently test embedded memories and, therefore, the rest of this section provides more details about them.

A. March Test Algorithms

A March test consists of a finite sequence of March elements [8]. A March element is a finite sequence of operations or primitives applied to every memory cell before proceeding to next cell [8]. 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). In summary, the notation of March test is described as follows:

↕	Addressing order can be either increasing or decreasing;
↑	Increasing memory addressing order;
↓	Decreasing memory addressing order

March algorithms are very easy to implement in either software or hardware. Table 2.4 [8] shows several relevant March algorithms reported in the literature.

Table 2: Irredundant March 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 C-	{ ↕(w0); ↑ (r0,w1);↑ (r1,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)}

w0 - Write 0 to a memory location;
w1 - Write 1 to a memory location;
r0 - Read 0 from a memory location;
r1 - Read 1 from a memory location;

B. Proposed Algorithm

This Algorithm is developed by modification in March Yalgorithm. The modified algorithm is able to detect Stuck at fault, Transition fault, coupling faults and also able to detect most Neighborhood pattern sensitive faults.

(↑wa↑ra) (↑wb↑rb) (↓wa↓ra)((↓wb ↓rb)

This type of the Data pattern will help to find out of all type of the fault such as Stuck-at fault, Transition fault, Coupling fault, Address decoder fault, Address decoder open fault, Retention fault, most Neighborhood pattern sensitive fault, linked faults and others fault occurring in deep submicron technology.

```
1.1   for (i = 0; i <= n-1; i = i+1)
        A [i] = ? ;          // Write Random Pattern
1.2   for (i = 0; i <= n-1; i = i+1)
        A [i] = ? ;          // Read the Pattern
2.1   for (i = 0; i <= n-1; i = i+1)
        A [i] = A [i]'       // Write Random Pattern
2.2   for (i = 0; i <= n-1; i = i+1)
        A [i] = ? ;          // Read the Pattern
3.1   for (j =n-1; i >= 0; j = i-1)
        A [j] = A [i]';      // Write Random Pattern
3.2   for (j =n-1; i >= 0; j = i-1)
        A [i] = ? ;          // Read the Pattern
4.1   for (j =n-1; i >= 0; j = i-1)
        A [j] = A [i],       // Write Random Pattern
4.2   for (j =n-1; i >= 0; j = i-1)
        A [j] = ? ;          // Read the Pattern
```

All types of the faults like Transition faults, address write and read faults, Oscillation faults, Linked fault Delay Faults with abrupt data change, structural faults to be detected by the proposed algorithm.

IV. SIMULATION RESULT

A. BIST with Memory Architecture

The design of the BIST with 64x32 memory is synthesized using the Leonardo Spectrum in 50 nm technology is shown in the following fig.

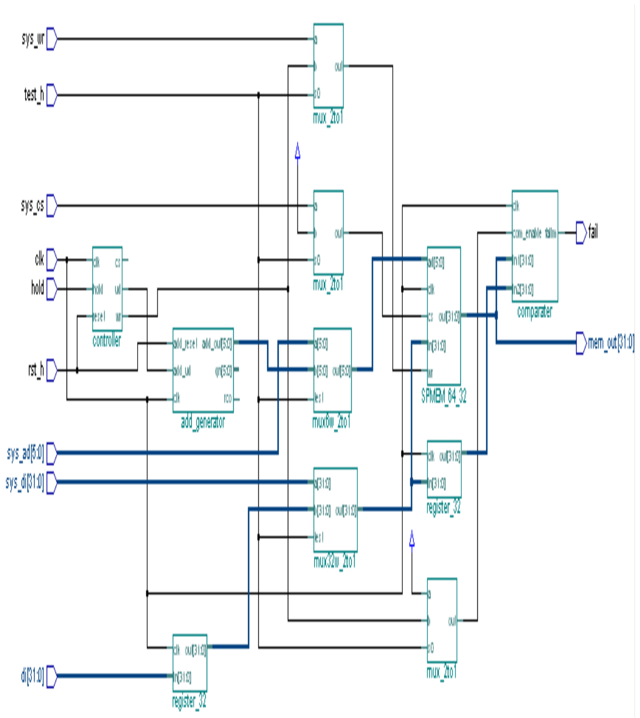

Fig.10: Design of BIST with 64x32 memory

B. MBIST in testing Mode

When the test_h signal is high then the MBIST is in test mode. The fail signal give the result related to the memory failure, if fail signal is high it means the memory affected by fault. A fail 0 tells about the memory correctness.

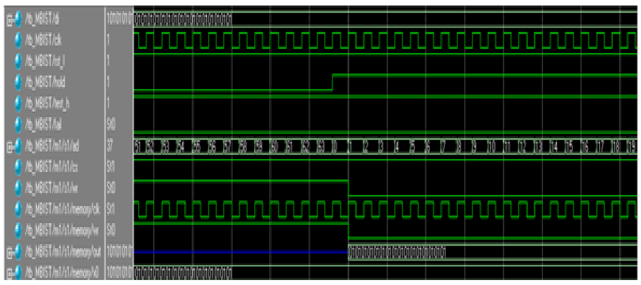

Fig. 11. Simulation result

V. CONCLUSION

Memory testing is very important but challenging. Memory BIST is considered the best solution due to various engineering and economic reasons. March tests are the most popular algorithms currently implemented in BIST hardware. Various implementation schemes for memory BISTs are presented and their trade-offs are discussed: A Hardwired-based BIST is fast and compact, whereas a Processor-based BIST cost near zero hardware overhead and very flexible. Different proposed innovations are also surveyed. Using Fault Coverage as our measure for test quality is revolutionary. Integrating diagnostic capabilities into BIST improves overall system robustness and chip yield. Automatic generation eases design efforts for test integration and help satisfying time-to-market requirements. Self-reparability is the key to fault tolerant and reliable circuit.

In conclusion, the future Memory BIST designs should be fast, small, efficient, robust, and flexible.

REFERENES

1. M. Abramovici, M. Breuer, and A. Friedman. Digital Systems Testing and Testable Design. IEEE Press, 1995.
2. S. M. Al-Harbi and S. K. Gupta. An Efficient Methodology for Generating Optimal and Uniform March Tests. In Proc. IEEE VLSI Test Symposium, pages 231–237, 2001.
3. M.L.Bushnell and V. D. Agrawal. Essentials of Electronic Testing. Kluwer Academic Publishers, 2000.
4. International SEMATECH. The International Technology Roadmap for Semiconductors (ITRS): 2001.
5. P. H. Bardell, W. H. McAnney, and J. Savir. Built-In Test for VLSI: Pseudorandom Techniques. John Wiley & Sons, Inc., New York, 1987
6. P. Mazumder and K. Chakraborty. Testing and Testable Design of High-Density Random-Access Memories. Kluwer Academic Publishers, 1996.
7. J. Saxena, K. M. Butler, V. B. Jayaram, S. Kundu, N. V. Arvind, P. Sreeprakash, and M. Hachinger. A Case Study of IR-Drop in Structured At-Speed Testing. In Proc. IEEE International Test Conference, pages 1098–1104, 2003.
8. A. J. van de Goor. Testing Semiconductor Memories: Theory and Practice. A.J. van de Goor, 1998.
9. Said Hamdioui: Linked Faults in Random Access Memories: Concept, Fault Models, Test Algorithms, and Industrial Results 0278-0070/04,IEEE- 2004.
10. I. Schanstra, D. Lukita, A. J. Van de Goor, K. Veelenturf and P. J van Wijnen, in Proc. International Test Conference (Washington DC, Oct., 1998), p. 872.
11. K. Zarrineh and S. J. Upadhyaya, Design, Automation and Test in Europe Conference (Munich, March, 1999), p. 708.
12. Sungju Park, Donkyu Youn, : Microcode-Based Memory BIST Implementing Modified March Algorithms. Journal of the Korean Physical Society April 2002
13. R. Rajsuman. Design and Test of Large Embedded Memories: An Overview. IEEE Design and Test of Computers, 18(3):16–27, May-June 2001.
14. Po-Chang Tsai, Sying-Jyan Wang and Feng-Ming Chang: FSM-Based Programmable Memory BIST with Macro Command. IEEE 2005.
15. Sying-Jyan Wang and Chen-Jung Wei: Efficient Built-In Self-Test Algorithm for Memory. 1081-773510 IEEE 2000.

AUTHORS PROFILE

Manoj Vishnoi received the M.Tech Degree in VLSI DESIGN from SRM University Chennai in 2010. He is working as Assistant professor in Department of ECE, SRM University NCR Campus modinagar since July 2010. His Research area is Low power VLSI, Memory DEvices.

Arun Kumar received the M.Tech Degree in VLSI DESIGN from SRM University Chennai in 2010. He is working as Assistant professor in Department of ECE, SRM University NCR Campus modinagar since July 2010. His Research area is Low power VLSI, Modeling of semiconductor devices, DSP.

Minakshi Sanadhya received the M.Tech Degree in VLSI DESIGN from SRM University Chennai in 2010. She is working as Assistant professor in Department of ECE, SRM University NCR Campus modinagar since July 2010. His Research area is Low power VLSI, Tunnel FET, DSP.