AN EFFICIENT AND LOW POWER SRAM TESTING USING CLOCK GATING

Sravani Kapavarpu#1 and Srinivas Cheedarla#2

#1Department of ECE, UshaRama College of Engineering Technology, Vijayawada, India
#2Assistant Professor, ECE, UshaRama College of Engineering Technology, Vijayawada, India

---

ABSTRACT: *Testing of memory devices in SOC is a main and important process in now days. Reliability and security plays vital role in SOC environment. Memory elements occupy almost 80 percent of whole area compare with other devices. An efficient and enhanced memory testing with at most secured low power algorithms are implemented in this concept. Main drawback of memory elements in SOC environment is because of tightly integration `in regarding density wise preformation. Due to that tight integration of memory, reliability issues come in to picture. A SOC failure occurs because of this memory soft and hard errors. Some old Built in Self test approaches are used to detect failure rates, but not much accurate and more power consuming processes are they. So, this thesis presents an efficient architecture of memory built in self test with variable configurations. The most flexible algorithms like MARCH C- and TLAPNPSF are used here for efficient memory testing. Along with that, more flexible architecture with energetic configuration of highest address constraint and the state machine design enables to reach the multiple range of memory measurements. Further this project is enhanced by using clock gating technique for further improvement of power. Clock gating technique is applied to ring counter to select the addresses with sophisticated mechanism. Here, the concept involved in clock gating technique is providing power supply to the block; in which that selected row is presented. No supply will be provided to the non-accessed rows.*

Keywords*:* MBIST; March C; TLAPNPSF; SRAM; ASIC

1. INTRODUCTION

Nowadays, the area engaged by hardware memories in System-on-Chip (SoC) is over almost 90%, and expected to increases up to 96% by 2020. Because all of those memories are very closely joined with more number of transistors causes 90% of overall faults in system on chips Thus, they concentrate the large majority of defects. Along with belligerent less nanometer scaling technology, blemish types are becoming more and more complex, various and may drip detection during fabrication test. If they are not treated adequately, the above trends will increase defect level, affect circuit quality dramatically and impact reliability, as undetected fabrication faults will be manifested as field failures. To cope with, the ability to guaranty a high quality test should be integrated in memory BIST, which is the mainstream test technology for embedded memories. Memory BIST generators can integrate a limited set of test algorithms[1][2] There are three memory test stimuli components: the test algorithm determining the operations performed in each memory cell and the instances they performed; the data used in these operations; and the sequence in which the memory addresses are visited by the test algorithm. Previous work comprises programmable BIST enabling test algorithm programmability [4-8] and data programmability [7][9], but no previous work exist concerning address sequence programmability. In the present paper we extend programmable BIST to incorporated address sequence programmability in addition to test algorithm programmability and test data programmability. Thus, all the components of memory test stimuli could be programmed in silicon, enabling testing unexpected failures during fabrication go/no go test, as well as comprehensive testing and diagnosis during failure analysis of customer returns; debug of new fabrication process or new memory design; and production ramp-up. The main challenge when implementing complete programmability of the address sequence used concerns the large amount of data that have to be programmed (here the complete memory address space) and the associated high hardware cost. We resolve this problem by adapting the transparent BIST scheme [10] in a way enabling storing the address sequence in the memory under test and using it for testing the memory. This paper talks about walking, marching and galloping pattern tests for RAM. Random access memory circuits are among some of the highly dense VLSI circuits that are being fabricated today. Since the transistor lines are very close to each other, RAM circuits suffer from a very high average number of physical defects per unit chip area compared with other circuits. This fact has motivated researchers to develop efficient RAM test sequences that provide good fault coverage. For testing today's high density memories traditional algorithms take too much test time. For instance GALPAT and WALKING I/O[5][6] require test times of order n^2 and $n^{3/2}$(where n is the number of bits in the chip). At that rate, assuming a cycle time of 100 ns, testing a 16Mbit chip would require 500 hours for an n^2 test and 860 seconds for an order $n^{3/2}$ test. Other older tests, such as Zero-One and Checkerboard, are of order n, but they have poor fault coverage. Due to the rapid progress in the very large scale integrated (VLSI) technology, an increasing number of transistors can be fabricated onto a single silicon die.

2. MARCH TEST ALGORITHMS

a) MARCH TEST ALGORITHMS

Based on the used memory fault models, memory test algorithms can be divided into four categories [9] as described below:
1. Traditional tests including Zero-One, Checkboard, GALPAT and Walking 1/0, Sliding Diagonal, and Butterfly [9]. 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].
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.

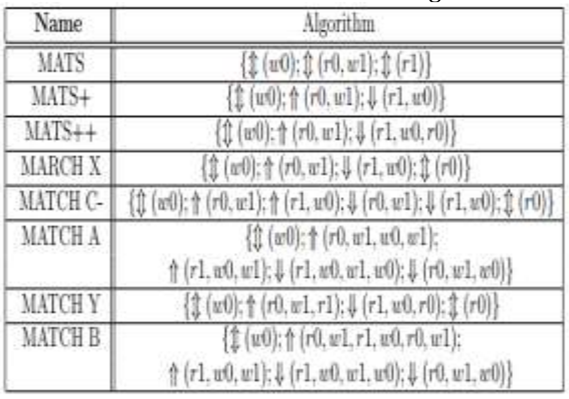
b) MARCH TEST NOTATION

A March test consists of a finite sequence of March elements [3]. A March element is a finite sequence of operations or primitives applied to every memory cell before proceeding to next cell [9]. 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 (m). 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: m Addressing order can be either increasing or decreasing; ⇑ Increasing memory addressing order; ⇓ Decreasing memory addressing order;

Table 1 Irredundant March Test Algorithms



Name	Algorithm
MATS	$\{\Updownarrow(w0); \Updownarrow(r0, w1); \Updownarrow(r1)\}$
MATS+	$\{\Updownarrow(w0); \Uparrow(r0, w1); \Downarrow(r1, w0)\}$
MATS++	$\{\Updownarrow(w0); \Uparrow(r0, w1); \Downarrow(r1, w0, r0)\}$
MARCH X	$\{\Updownarrow(w0); \Uparrow(r0, w1); \Downarrow(r1, w0); \Updownarrow(r0)\}$
MATCH C-	$\{\Updownarrow(w0); \Uparrow(r0, w1); \Uparrow(r1, w0); \Downarrow(r0, w1); \Downarrow(r1, w0); \Updownarrow(r0)\}$
MATCH A	$\{\Updownarrow(w0); \Uparrow(r0, w1, w0, w1);$ $\Uparrow(r1, w0, w1); \Downarrow(r1, w0, w1, w0); \Downarrow(r0, w1, w0)\}$
MATCH Y	$\{\Updownarrow(w0); \Uparrow(r0, w1, r1); \Downarrow(r1, w0, r0); \Updownarrow(r0)\}$
MATCH B	$\{\Updownarrow(w0); \Uparrow(r0, w1, r1, w0, r0, w1);$ $\Uparrow(r1, w0, w1); \Downarrow(r1, w0, w1, w0); \Downarrow(r0, w1, w0)\}$

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

3. SRAM USING CLOCK GATING TECHNIQUE

This section describes PJMEDIA's implementation of delay buffer.

a) Delay Buffer

Delay buffers are used to store the data in temporary manner. Like Random Access memory, these delay buffers accessed using ring counters constructed by D flipflops. Buffer, that is it will latency the frame recovery by some interlude so that caller will get incessant frame from the buffer. This will be perfectly useful when the accessing operations are not repeatedly interleaved, for example when caller performs burst of put() operations and then followed by burst of operations. With using this delay buffer, the buffer will put the burst frames into a buffer so that get() operations will always get a frame from the buffer (assuming that the number of get() and put() are matched).The sample buffer diagram is shown in the figure. The buffer is variable, that is it continuously learns the optimal delay to be applied to the audio flow at run-time. Once the optimal delay has been learned, the delay buffer will apply this delay to the audio flow, expanding or shrinking the audio samples as necessary when the actual audio samples in the buffer are too low or too high.

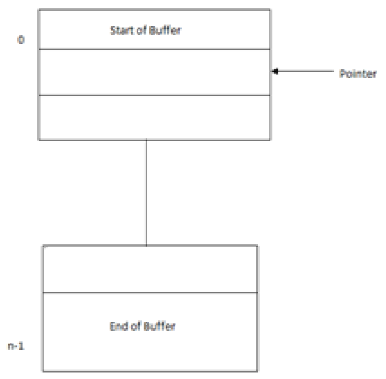


Fig 1: Delay Buffer

b) Ring Counter
A ring counter is a type of counter composed of a circular shift register. The output of the last shift register is fed to the input of the first register.

There are two types of ring counters:

A straight ring counter or Overbeck counter connects the output of the last shift register to the first shift register input and circulates a single one (or zero) bit around the ring. For example, in a 4-register one-hot counter, with initial register values of 1000, the repeating pattern is: 1000, 0100, 0010, 0001, 1000... . Note that one of the registers must be pre-loaded with a 1 (or 0) in order to operate properly.

A twisted ring counter (also called Johnson counter or Moebius counter) connects the complement of the output of the last shift register to its input and circulates a stream of ones followed by zeros around the ring. For example, in a 4-register counter, with initial register values of 0000, the repeating pattern is: 0000, 1000, 1100, 1110, 1111, 0111, 0011, 0001, 0000... .

If the output of a shift register is feed back to the input. A ring counter results the data pattern contained within the shift register will recirculate as long as clock pulses are applied. For example, the data pattern will repeat every four clock pulses in the figure 2. However, we must load a data pattern. All 0's or all 1's doesn't count.

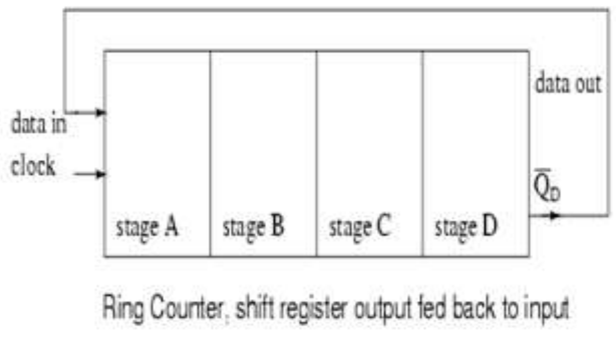


Fig 2: Ring counter using shift register

Loading binary 1000 into the ring counter, above, prior to shifting yields a viewable pattern. The data pattern for a single stage repeats every four clock pulses in our 4-stage example. The waveforms for all four stages look the same, except for the one clock time delay from one stage to the next as represented in figure 3.

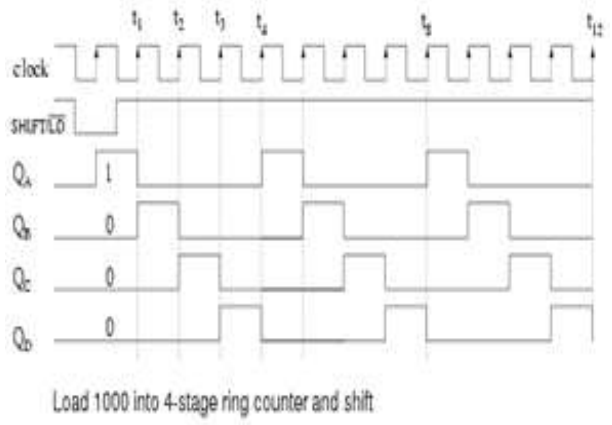


Fig 3: Load 1000 into 4-stage ring counter and shift

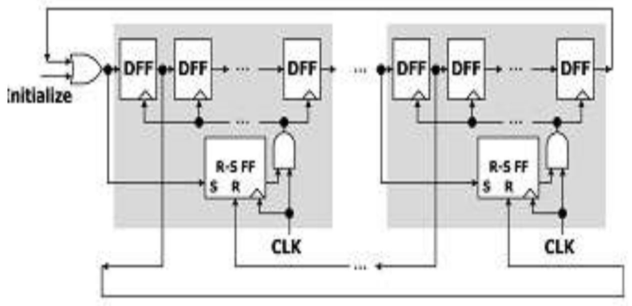


Fig 4: Ring Counter with SR Flip flop

The block diagram shows the power controlled Ring counter. First, total block is divided into two blocks.
Each block is having one SR FLIPFLOP controller which is shown as in fig 4.

IV. RESULT:MARCHC

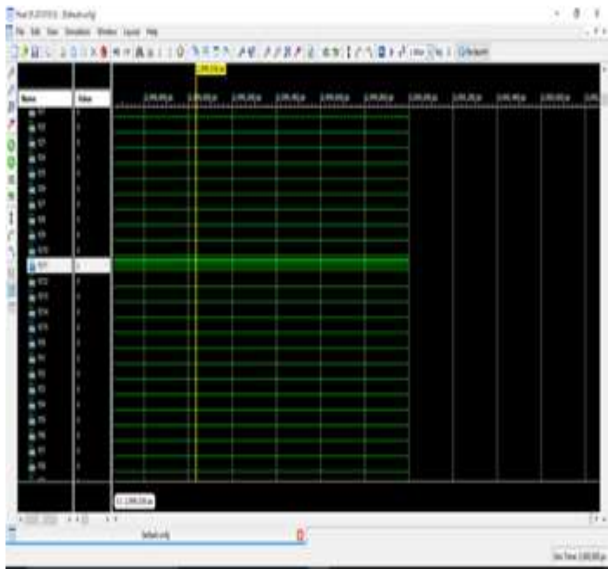


Fig 5: Simulation Result for MARCH C Algorithm

Fig 5 Shows the Simulation output of MARCH C Algorithm. This Algorithm shows the faults for which we are giving inputs from f00 to f315 as '0' value if any faults occurred then the value will be'1' as shown in output representation.

TLAPNPSF

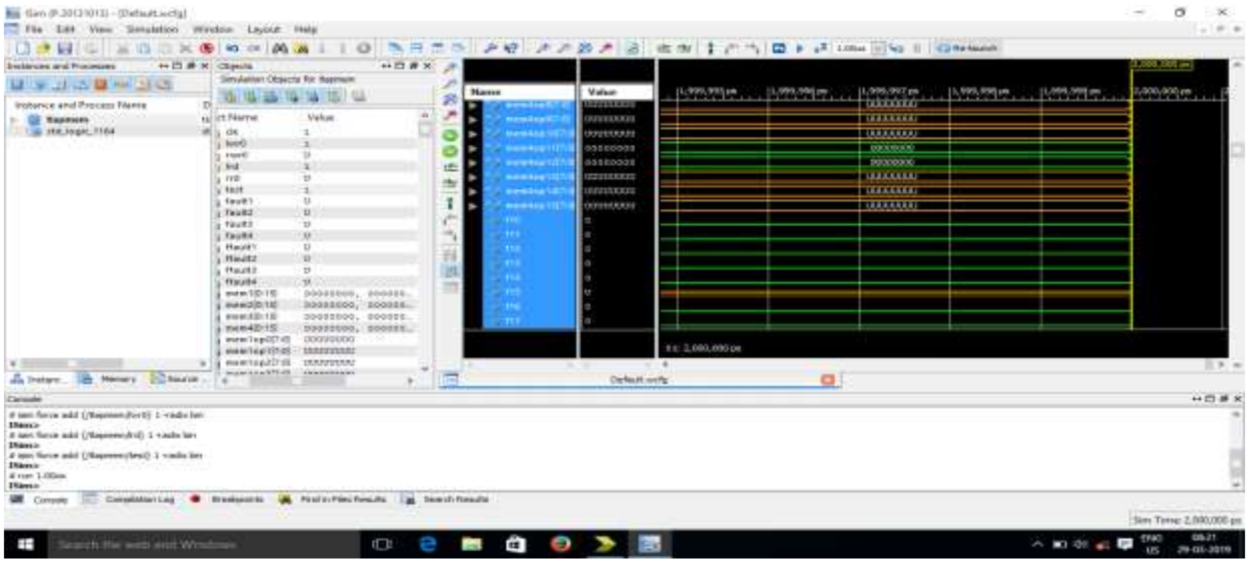


Fig 6: Simulation Result for TLAPNPSF Algorithm

Fig 6 shows the Simulation result for TLAPNPSF Algorithm which recover the faults in neighborhood cells in triangular type model The inputs we are giving same as MARCH C and the clock is given as the value'1' faults are shown in two ways either right oriented as well as in left oriented cells remaining cells must be '0'.By this We can detect the Faults very easily.

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 Defect Coverage not Fault Coverage as our measure for test quality is revolutionary. Clock gating and 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-repairability is the key to fault-tolerant and reliable circuit. In conclusion, the future Memory BIST designs should be fast, small, efficient, robust, and flexible.

VI. REFERENCES

[1] Erwing R.Sanchez and Maurizio Rebaudengo,"ANovel Access Scheme for online Test in RFID Memories",IEEE -2011.
[2] J.McDonnell, J. Waters, H. Balinsky, R. Castle, F. Dickin, W. W. Loh, and K. Shepherd, "Memory spot:
A labeling technology," Pervasive Computing, IEEE, vol. 9, no. 2, pp. 11–17, april-june 2010.
[3] E. Welbourne, L. Battle, G. Cole, K. Gould, K. Rector, S. Raymer,M. Balazinska, and G. Borriello, "Building the internet of things using rfid: The rfid ecosystem experience, Internet Computing, IEEE, vol. 13, no. 3, pp. 48–55, may-june 2009.
[4] EPCGlobal, EPC radio-frequency identity protocols Class-1 Generation-2 UHF RFID air interface Version 1.2.0, Oct. 2008.
[5] T. Cheng and L. Jin, "Analysis and simulation of rfid anti-collision algorithms," in Advanced Communication Technology, The 9th International Conference on, vol. 1, 2007, pp. 697–701.
[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[7] R. Dekker, F. Beenaker, L.Thijssen, "A realistic fault model and test algorithm for static random access memories", IEEE Trans. Computer-Aided Design, vol, 9, 99.567-572, june 1990.
[8] J. van de Goor. Testing Semiconductor Memories: Theory and Practice.A.J.vandeGoor,1998.
[9] Verilog Digital System Desigh, 2nd Edition, Zainalabedin Navabi, Tata McGraw Hill, 2008 .
[10] A.J.Van de Goor, Testing Semiconductor Memories, Theory and Practice, John Wiley & Sons, Chichester, England, 1991.