DOI: 30.3425/ijearst.41.93.5471 https://ijearst.co.in/

ENGINEERING IN ADVANCED RESEARCH SCIENCE AND TECHNOLOGY

ISSN 2386-7876
Vol.01, Issue.03
March-2026
Pages: 391-396

# AN EFFECTIVE ARCHITECTURE OF MEMORY BUILT-IN SELF-TEST FOR WIDE RANGE OF SRAM

[1] **V.Harika**, [2] **Vanarasi Sri Satya Sai Naga Pushpa Latha**, [3] **M.Sowjanya Lakshmi**, [4] **D.Jahnavi sri satya**, [5] **S.Naveen babu**

[1]M.Tech,Assistant Professor, Dept of E.C.E, BVCITS, Batlapalem, Amalapuram, AP

[2,3,4,5]B. Tech, Dept of E.C.E, BVCITS, Batlapalem, Amalapuram, AP

**ABSTRACT:** Testing semiconductor memories is increasingly important today because of the high density of current memory chips. Most of the System-on-Chip (SoC) area covered by embedded memories. As these memories are very tightly integrated, consists majority of defects in soc. Detection of such a complex and diverse faults during fabrication is not possible. Hence leads to failure of soc in field. Usage of test algorithms may increase the coverage of complex faults, but unexpected failures can't be covered by these algorithm. Providing possibility of choosing testing algorithms before using in SoC is very important. Programmable BIST approaches, allowing selecting after fabrication a large variety of memory tests, are therefore desirable, but may lead on unacceptable area cost. In this paper we investigate on the various functional fault models present for today's memory technology and discuss about the ways and means to detect these faults. This project presents an effective architecture of MBIST for SRAM type with different configurations is proposed for not only ensuring high ability of detecting memory faults supported by the most popular algorithms namely MARCH C- and TLAPNPSF. Besides, architecture with dynamic configuration of highest address parameter and the state machine design enables to reach the wide range of memory depth. To ensure all fault cases and all configurations are covered, an automatically verification environment is designed to make progress conveniently. 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:** System-on-Chip, MBIST; MarchC; TLAPNPSF; SRAM; ASIC.

**INTRODUCTION:** Nowadays, the area occupied by embedded memories in System-on-Chip (SoC) is over 90%, and expected to rise up to 94% by 2015. As those memories are very tightly integrated with large number of transistors causes 90% of overall faults in system on chips Thus, they concentrate the large majority of defects. In addition, with aggressive nanometer scaling, defect types are becoming more complex and diverse and may escape 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 (see for instance [1][2][3]). Thus, only the test algorithms selected during the design phase can be used after fabrication. However, fixing the memory test algorithms during the design phase is not a good strategy as unexpected failures may be discovered after production. Also, integrating pre-emotively a large number of test algorithms in the BIST generator will result in large area cost. Thus, programmable memory BIST enabling selecting the memory, y test stimuli in silicon and testing the memory for a wide variety of faults is becoming mandatory. This flexibility has to be achieved at low area cost, to make the approach attractive for real products. Also, the flexibility offered by programmable BIST is highly important for thorough screening inspection, failure analysis of customer returns, debug of a new fabrication process or a new memory design, and production ramp-up, since the most challenging issue in these processes is to detect and/or diagnose unexpected failures. 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 [10-16] 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.

## 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.

## 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;

| 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 | $\{\Updownarrow (w0); \Uparrow (r0, w1, r1, w0, r0, w1);$ $\Uparrow (r1, w0, w1); \Downarrow (r1, w0, w1, w0); \Downarrow (r0, w1, w0)\}$ |

Table 1 Irredundant March Test Algorithms

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;

**MARCH TEST ALGORITHMS:** Table lists several relevant March algorithms reported in the literature. Table gives the fault coverage and the operation count of these March algorithms, which are also called irredundant algorithms (by removing any operation from the test, the targeted fault coverage will be reduced). To generate custom March algorithms for improved defect coverage in new process technologies, an effective methodology was proposed in [2]. March algorithms are very easy to implement in either software or hardware. A piece of pseudo-code for the MATS+ algorithm is given to demonstrate the basic test procedures. In the code shown below, n is the total number of bits of the memory (bit-oriented memory) and Addr[i] points to the ith memory address for read or write. Line 1 runs the first March element of MATS+ algorithm m (w0). Since the address sequence can be either up or down, here we use an up address sequence. Line 2 to 5 run the second March element ⇑ (r0, w1) with the up address sequence. Line 6 to 9 implement the third March element ⇓ (r1, w0) with the down address sequence. If any data mismatch happened during the test (line 3 and 7) the program will stop and return fail. Otherwise, it will return success after all March elements are finished

| | Fault Coverage | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Algorithm** | **SAF** | **AF** | **TF** | **CF in** | **CF id** | **CF dyn** | **SCF** | **Linked Faults** | **Oper. Count** |
| MATS | All | Some | | | | | | | 4.n |
| MATS+ | All | All | | | | | | | 5.n |
| MATS++ | All | All | All | | | | | | 6.n |
| MARCH X | All | All | All | All | | | | | 6.n |
| MARCH C- | All | All | All | All | All | All | All | | 10.n |
| MARCH A | All | All | All | All | | | | All linked CFids, some CFins linked with CFids | 15.n |
| MARCH Y | All | All | All | All | | | | All TFs linked with CFins | 8.n |
| MARCH B | All | All | All | All | | | | All linked CFids, all TFs linked with CFids or CFins, some CFins linked with CFids | 17.n |

Table:2 Irredundant March Test Summary

9 implement the third March element ⇓ (r1, w0) with the down address sequence. If any data mismatch happened during the test (line 3 and 7) the program will stop and return fail. Otherwise, it will return success after all March elements are finished. Characteristics of March Algorithms March-based memory test algorithms have several important characteristics:

• Up (down) address sequence must be the exact reverse down (up) sequence, however its internal order is irrelevant. For example, if a 3 bits up address sequence is {0, 5, 2, 3, 7, 1, 4, 6}, then the down sequence must be {6, 4, 1, 7, 3, 2, 5, 0}.

• Most March algorithms are only a simple combination of several March elements (e.g., ⇑ (r0, w1) is a March element). By analyzing the March algorithms shown in Table 2.5, it can be observed that the background pattern during execution can be inferred by the previous operation. For example, a read operation infers the same background data used in the last operation. Similarly, a write operation infers the reversed background data used in the last operation. For example, the first operation of the March C- test shown in Table 2.4 is w0. The next operation is read (must be r0) and the following operation is write (must be w1). Based on this observation, one can reduce the number of March elements and the complexity of their implementation. For Match C-, only three March elements are needed: (w), (r, w), (r). The total number of March elements for the most practical March algorithms is less than ten.

## SRAM USING CLOCK GATING TECHNIQUE:

MEMORY ORGANIZATION: This section describes PJMEDIA's implementation of delay buffer. Delay buffer works quite similarly like a fixed jitter buffer, that is it will delay the frame retrieval by some interval so that caller will get continuous frame from the buffer. This can be useful when the operations are not evenly 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 buffer is adaptive, 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.

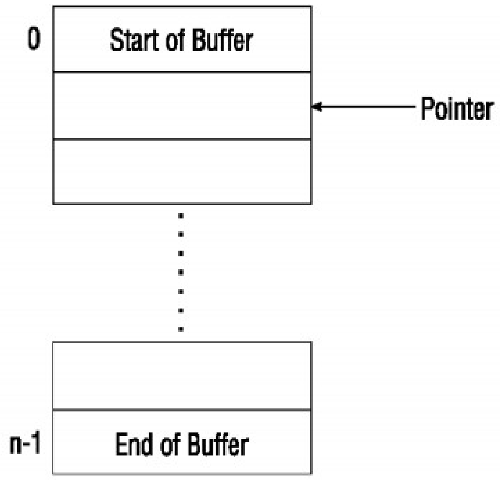
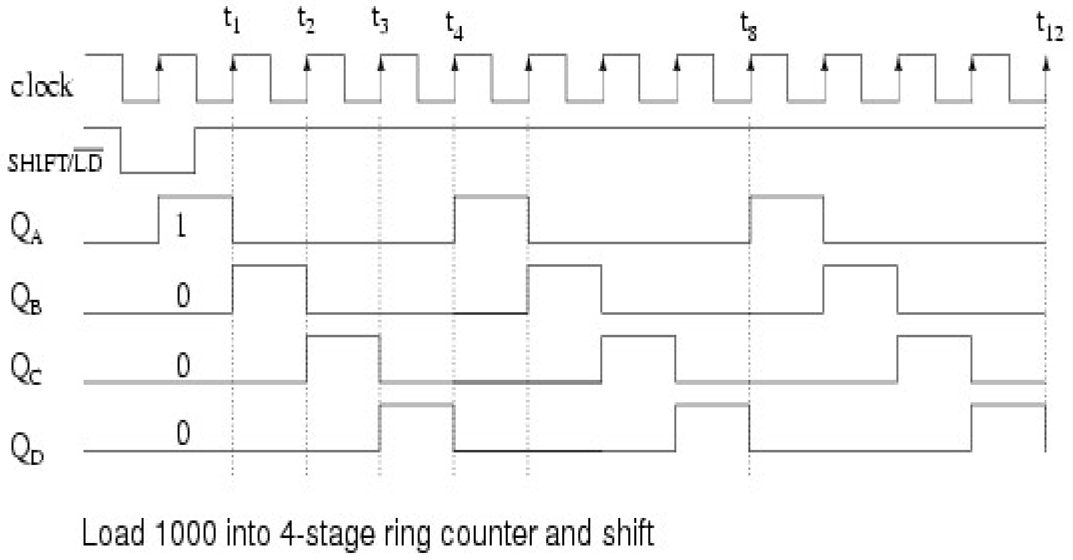


Fig1: Buffer

Loading binary **1000** into the ringcounter, 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. See figure below.



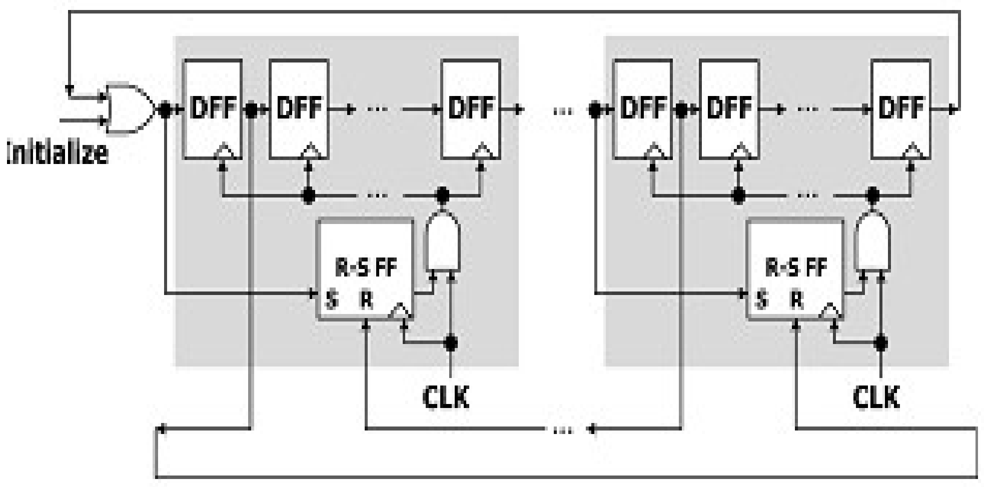
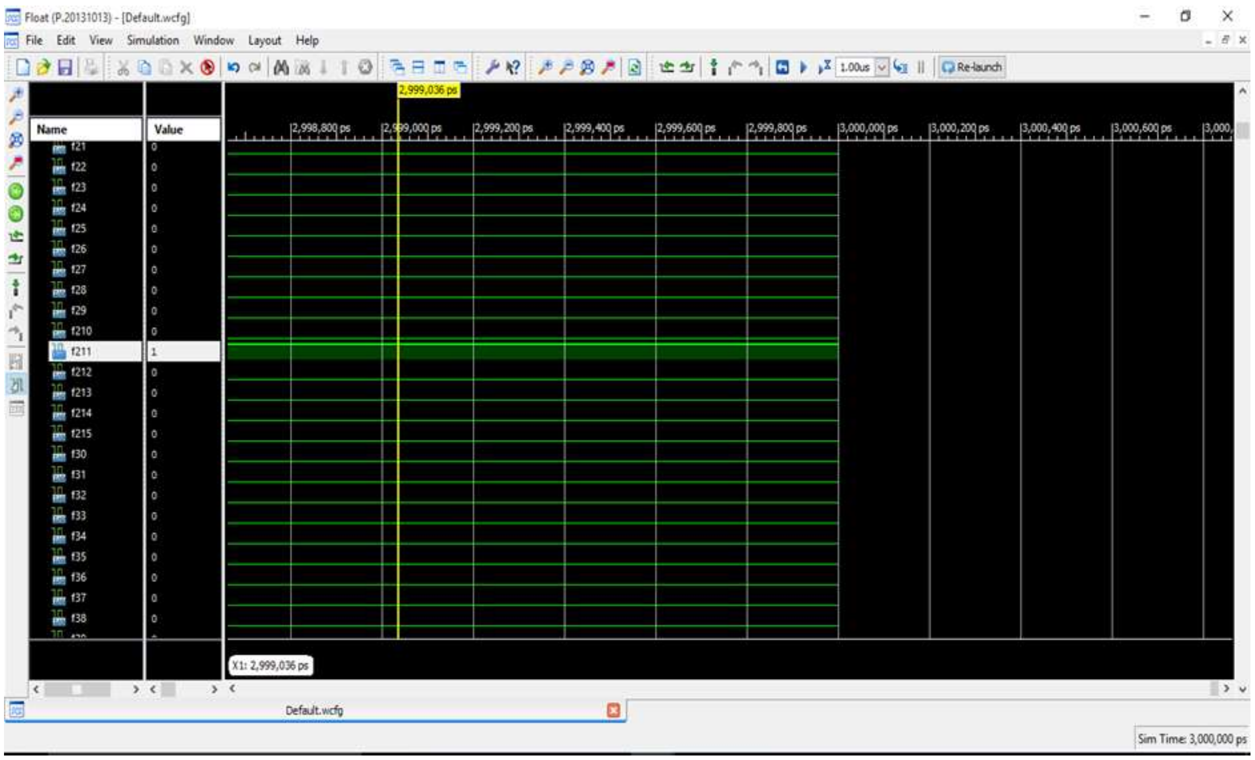


**Fig :2** Ring Counter With SR Flip-Flops

The above block diagram shows the power controlled Ring counter.First, total block is devided into two blocks. Each block is having one SR FLIPFLOP controller

**RESULT:**



**Figa: Simulation result**

**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.

**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.