An Accumulator based Compaction Scheme for Online BIST of RAMS

Md.Shazia Fatima[1] **Harish[2]** **P. Vamshidhar Reddy[3]**
zsha194@gmail.com[1] vamshidhar227@gmail.com[3]
[1]PG Scholar Dept of ECE, Sreenidhi Institute of Science and Technology,Yamnampet, Ghatkesar, Hyderabad, Telangana.
[2]Assistant Professor, Dept of ECE, Sreenidhi Institute of Science and Technology,Yamnampet, Ghatkesar, Hyderabad, Telangana.
[3]Assistant Professor, Dept of ECE, Sreenidhi Institute of Science and Technology,Yamnampet, Ghatkesar, Hyderabad, Telangana.

***Abstract:* — Transparent built-in self test (BIST) schemes for RAM modules assure the preservation of the memory contents during periodic testing. Symmetric transparent BIST skips the signature prediction phase required in traditional transparent BIST schemes, achieving considerable reduction in test time. In symmetric transparent BIST schemes proposed to date, output data compaction is performed using either single-input or multiple input shift registers whose characteristic polynomials are modified during testing. In this project the utilization of accumulator modules for output data compaction in symmetric transparent BIST for RAMs is proposed. It is shown that in this way the hardware overhead, the complexity of the controller, and the aliasing probability are considerably reduced.**

***Index Terms*—Online testing, random access memories (RAMs), self testing.**

I. INTRODUCTION

With the advent of deep-submicron VLSI technology, the memory density and capacity is growing. The clock frequency is never higher. The dominant use of embedded memory cores along with emerging new architectures and technologies make providing a low cost test solution for these on-chip memories a very challenging task. Built-in self-test (BIST) has been proven to be one of the most cost-effective and widely used solutions for memory testing

For the testing of embedded RAMs, march algorithms outperform competitive schemes, since they result in simple, yet effective, testing scenarios [5]. A march algorithm comprises a series of march elements.

Traditional march algorithms, start with an initial write-all-zero phase, where all the RAM cells are set to 0 in order to ensure that the final signature in the output compactor. Periodic testing is discerned into start-up testing and testing during normal operation. Start-up testing is performed during the start-up of the system and resembles manufacturing testing. In testing during normal operation, the RAM normal operation is stalled (i.e., set out of normal operation), tested and then given back to operation. This kind of testing is applied to circuits where it is difficult and/or impractical to shut down the system since the contents of the RAM cannot be lost. In this kind of testing, traditional march tests cannot be applied since (due to the initial write-all zero phase) the contents of the RAM cells before the test are lost. In order to confront the previously mentioned problems, transparent built-in self test (BIST) was proposed by Nicolaidis [1]; in a transparent BIST algorithm, the initial write-all-zero phase is skipped; instead, a signature prediction phase is issued that precedes the normal march series.

This paper is organized as follows. In Section II Literature Survey, Section III, a review of the march algorithms (traditional, transparent, and symmetric transparent) is given. In Section IV, the proposed accumulator-based compaction scheme for symmetric transparent BIST (MISR) is introduced and exemplified for the case of word-organized RAMs. Results in Section V, and in Section VI we conclude this paper.

II. LITERATURE SURVEY.

Fabrication anomalies in the IC manufacturing process may cause some circuits to behave erroneously [3]. Manufacturing test helps to detect physical defects (e.g., shorts or opens) prior to delivering the packaged circuits to end-users. Once a defective chip has been detected, comprehensive defect screening through fault

diagnosis is required to adjust the manufacturing process and accelerate the yield learning curve [4].

The physical defects lead to faulty behaviors that can be detected by parametric tests for chip pins and tests for functional blocks [2]. Parametric tests include DC tests (such as voltage, leakage test and output drive current test) and AC tests (setup and hold time tests and propagation test). These tests are usually technology-dependent and can be done without any understanding of the chip functionality. The test for functional blocks involves modeling manufacturing defects at a certain level of design abstraction, such as behavioral level, register-transfer level (RTL), gate level or transistor level. Fault models based on gate level net lists are technology-independent and over time have been proven to be very efficient for testing digital circuits. Basic fault models for gate level testing are stuck-at, bridging and delay fault models. The single stuck-at fault model is the most popular fault model in digital system testing and is based on the assumption that a single node (line) in the structural net list of logic gates can be stuck to a logic value 0 (SA0) or 1 (SA1). The test for functional blocks determines whether the manufactured chip behaves as designed and because the gate count keeps on growing, the testing time for functional blocks is also increasing. Since the time a chip spends on an expensive tester directly influences the production cost, reducing the testing time of functional blocks is an essential task, which needs to be accomplished in order to lower the cost associated with manufacturing test.

The test of functional blocks can further be divided into structural test and functional test. If the test depends on the net list structure of the design then it is called structural test. Based on the targeted fault models (e.g., stuck-at), automatic test pattern generation (ATPG) tools generate test sets which sensitize the fault and propagate its effects to observation points (e.g., primary outputs). Functional test programs, on the other hand, generate a set of test patterns to verify the functionality of each component in the circuit. Because functional test is an exhaustive test method, testing time is prohibitively large for combinational logic blocks, which makes it infeasible for complex digital systems [2]. One exception is the test of semiconductor memories due to their regularity. Since the cells in a memory block have identical structure and they are not related one to each other, and because memory operations are simple (read and write), exhaustive functional test becomes tractable.

III. MARCH ALGORITHMS

Traditional Algorithm

For example, in fig. 3.1(a) the march-c algorithm consist of 6 March elements denoted by M0 to M5 [21]. In this fig.3.1 increasing addressing order is denoted by ⇑(it can be any arbitrary addressing order) and decreasing addressing order is denoted by a ⇓ (it is the inverse of ⇑). A traditional March test algorithm consists of $0 \le i \le n$ elements. Each element consist of zero or more write operations, denoted by w_0/w_1 means that 0/1 is written to the RAM(memory)cell, and zero or more read operations are denoted by r_0/r_1 means that 0/1 is expected to be read from the RAM(memory)cell.

M_0.	⇑ (w_0);	⇑ (r_a); ⇑ $((r_a)^c)$; ⇓ (r_a); ⇓ $((r_a)^c)$; ⇓ (r_a);	⇑ $((r_a)^c)$;
M_1.	⇑ (r_0, w_1);	⇑ (r_a, w_a^c);	⇑ (r_a, w_a^c);
M_2.	⇑ (r_1, w_0);	⇑ (r_a^c, w_a);	⇑ (r_a^c, w_a);
M_3.	⇓ (r_0, w_1);	⇓ (r_a, w_a^c);	⇓ (r_a, w_a^c);
M4.	⇓ (r_1, w_0);	⇓ (r_a^c, w_a);	⇓ (r_a^c, w_a);
M_5.	⇓ (r_0);	⇓ (r_a);	⇓ (r_a);
	(a)	(b)	(c)

Fig.1. C- march algorithm: (a) original version; (b) transparent version; and (c) symmetric transparent version.

Transparent BIST Algorithms.

Traditional march algorithms erase the memory contents prior to testing, therefore, they do not serve as good platforms for periodic BIST. Nikolaidis proposed the concept of transparent BIST where the initial W_0 phase is bypassed, and a signature prediction 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. The transparent version of the C-algorithm is shown in Fig.1.b. The notation for the transparent versions of the algorithms differs slightly from the one used in traditional 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. Their meanings are as follows.

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).

$\mathbf{r_a^c}$ - Read the contents of a word of the RAM, expecting to read the complement of the initial contents of the RAM word.

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

$\mathbf{w_a}$ - 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.

$\mathbf{w_a^c}$ - 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.

By default, the data driven to the compactor with the $(r_a)^c$ operation are identical to the data driven by the r_a^c. The importance of the $(r_a)^c$ 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 r_a^c operation we invert these contents prior to driving them to the compactor.

It has been observed that, 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 (M0) 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. Traditional transparent BIST has the disadvantage that the signature prediction phase adds up to the total testing time with a percentage of (more than) 30% [28]. In order to confront this problem, Yarmolik *et al.* introduced the concept of symmetric transparent BIST, which is explained in the next subsection.

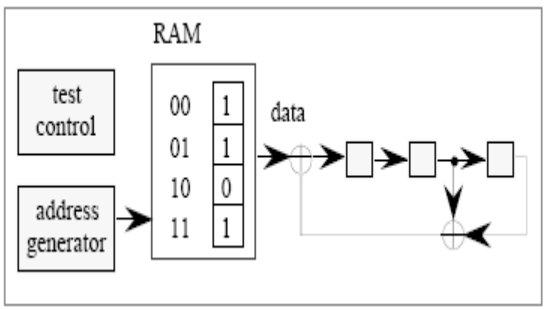


Fig2.: Example for transparent BIST.

Symmetric Transparent BIST:

Since March tests scan the complete memory several times, test time becomes an issue of growing importance with increasing memory densities. In particular, transparent BIST for maintenance purposes may become infeasible, because it requires a considerable amount of extra time to compute a reference result each time before it is applied. Figure illustrates this for a small example memory and a transparent version of the MATS+ algorithm. Following the notations defined in following table the original MATS+ algorithm can be written as { c(w0);•(r0, w1); •(r1, w0)}. It is transformed into a transparent test by deleting the initialization phase and replacing read and write operations with fixed „0“ or „1“ values by read and write operations with the appropriate variable values. For the transparent version {•(ra, wac); •(rac, wa)} a BIST session proceeds in two phases changing from increasing to decreasing address order. After each read operation the obtained data are fed into the signature analyzer and at the end of the second phase the final signature has to be compared to a reference signature REF. This reference signature, however, depends on the memory contents and must be computed in an extra „signature prediction“ phase before the test can be started. In general, the signature prediction phase basically consists of all read-operations of the complete test. For the transparent MATS+ algorithm it is described by {•(ra); •((ra)c)} and requires an extra time of 2n. The time for a complete maintenance test based on MATS+ is 6n, which implies that one third of the test time is required for signature prediction. This ratio is similar for any of the known transparent BIST algorithms.

IV. An Accumulator – based compaction of the responses in symmetric transparent bist

The accumulator-based response compaction scheme proposed in this project, stems from the following two observations.

1) *Observation 1:* If the march algorithm is symmetric (as in the case of symmetric transparent BIST) then the number of r_a elements equals the number of ra^c elements plus the number of $((r_a)^c)$ elements (without

taking into account the addressing order of the march element).

2) *Observation 2:* The accumulator-based compaction of the responses holds the *order-independent* property (i.e., the final signature is independent of the order of the incoming vectors). Observation 2 stems directly from the permutational property of the addition operation.

Accumulator-based compaction for symmetric transparent BIST for the case of word-organized memories is based on Lemma 1.

Lemma 1: If a symmetric transparent march algorithm is applied to a word-organized memory whose word length is n and the responses are captured in an n-stage accumulator comprising a 1's complement adder (starting from the all-0 state), then the final content of the accumulator is equal to the all-1 state.

Proof: Let_ be the number of elements of the march algorithm; since the algorithm is symmetric, the total number of r_a elements is equal to the total number of r_a^c plus the number of $((r_a)^c)$ elements. Therefore, for every vector a driven to the inputs of the accumulator, its complement ac is also driven to the inputs of the accumulator exactly once. But

$$a + a^c = 2^n-1$$

Furthermore, for 1's complement addition it holds that the sum of two numbers A and B is given by (A-1 + B(mod 2^n-1) + 1), therefore the sum of 2^n-1 and 2^n-1 is $(2^n-1-1+2^n-1) \bmod (2^n-1)+1 = 2^n-2+1$. Therefore it is trivial to show (by induction) that the equation below holds for any value of i.

$$\sum_i i \times (2^n-1) = 2^n-1$$

Taking the above two equations into account that the addition operation is per -mutative, we have the proof.

For example, let us consider the 4-word 3-bit RAM presented in the following Fig3.(a). The outputs of the memory are driven to an n 3-stage accumulator comprising a 1's complement adder, Fig3.(b). For the implementation of the r_a^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 in Fig3.. Therefore, the inverse of the read vector appears at the outputs the adder/subtractor and applied to the RAM inputs.

Address and Memory Contents, took as an example for Showing the Compression using Accumulator

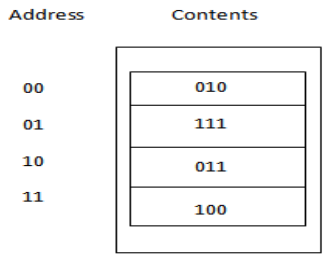


Fig: 3. (a). Accumulator-based compaction in word-organized memories.

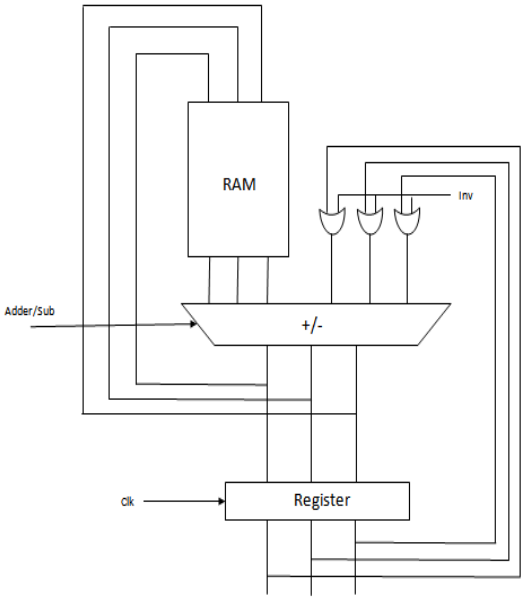


Fig 3.(b) Micro Architecture for Compression & Accumulator-based compaction in word-organized memories.

Writing the data is as it is, if we want to write normal data. If we want to write data in its complemented form, we need to enable the inv signal, so as to make the output of the OR gates as all 1's and

the Subtractor is activated, so as to get the outputs in its complemented form.

In the similar fashion, the entire algorithm is repeated; in the end/completion of all the steps of the algorithm, we will find all the 1s in the accumulator (or in register).

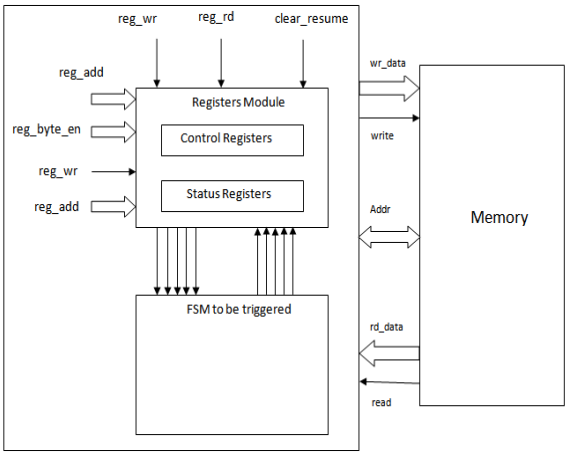
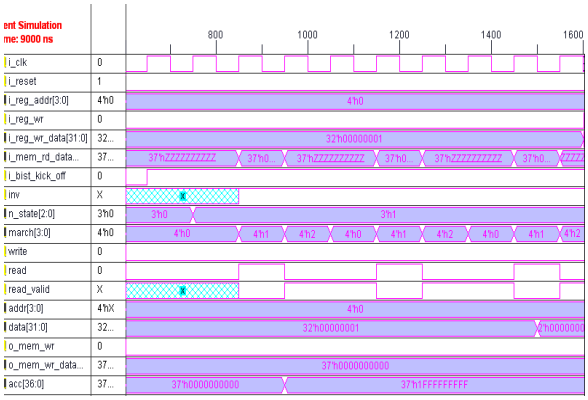


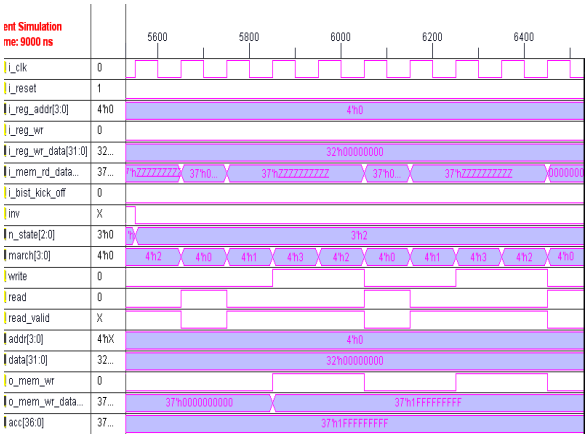
Fig: 4.Macro Architecture (or Top-Level Architecture for MBIST)

V.RESULTS

State M0 Report.



State M1 Report.



VI. Conclusion

In this paper, we have proposed the utilization of accumulators for the compaction of the responses in MBIST Architecture. The proposed scheme presents lower hardware overhead and requires less complicated control compared to the scheme proposed in previous versions, (where the MISR is used for compaction), therefore, it may prove a viable solution for periodic testing of RAMs embedded into current VLSI chips.

REFERENCES

[1] M. Nicolaidis, “Theory of transparent BIST for RAMs,” IEEE Trans. Comput., vol. 45, no. 10, pp. 1141–1156, Oct. 1996.

[2] M. Nicolaidis, “An efficient built-in self test for functional test of embedded RAMs,” in Proc. 15th Symp. Fault Tolerant Comput., Jun. 1985, pp. 118–123.

[3] A. Castro, M. Nicolaidis, P. Lestrat, and B. Courtois, “Built-in self test for multi-port RAMs,” presented at the ICCAD, Santa Clara, CA, Nov. 1991.

[4] A. J. van de Goor, Testing Semiconductor Memories, Theory and Practice. Chichester, U.K.: Wiley, 1991.

[5] A. J. van de Goor, “Using march tests to test SRAMs,” IEEE Des. Test Comput., vol. 10, no. 1, pp. 8–14, 1993.

[6] A. J. van de Goor and C. A. Verruijt, “An overview of deterministic functional RAM chip testing,” ACM Comput. Surveys, vol. 22, no. 1, pp. 5–33, Mar. 1990.

[7] M. Marinescu, “Simple and efficient algorithms for functional RAM testing,” in Proc. IEEE Int. Test Conf., 1982, pp. 236–239. [8] V. N. Yarmolik, S. Hellebrand, and H.-J. Wunderlich, “Symmetric transparent BIST for RAMs,” presented at the DATE, Munich, Germany, Mar. 1999.

[9] V. N. Yarmolik, I. V. Bykov, S. Hellebrand, and H.-J. Wunderlich, “Transparent word-oriented memory BIST based on symmetric march algorithms,” in Proc. Eur. Dependable Comput. Conf., 1999, pp. 339–350.

[10] I. Voyiatzis, “Test vector embedding into accumulator-generated sequences: A linear-time solution,” IEEE Trans. Comput., vol. 54, no. 4, pp. 476–484, Apr. 2005.

[11] A. Stroele, “BIST patter generators using addition and subtraction operations,” J. Electron. Test.: Theory Appl., vol. 11, pp. 69–80, 1997.

[12] R. Dorsch and H.-J. Wunderlich, “Accumulator-based deterministic BIST,” in Proc. Int. Test Conf., 1998, pp. 412–421.

[13] I. Voyiatzis, A. Paschalis, D. Gizopoulos, N. Kranitis, and C. Halatsis, “A concurrent built-in self-test architecture based on a self-testing RAM,” IEEE Trans. Reliab., vol. 54, no. 1, pp. 69–78, Mar. 2005.

[14] W. L. Wang, K. J. Lee, and J. F. Wang, "An on-chip march pattern generator for testing embedded memory cores," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 9, no. 5, pp. 730–735, Oct. 2001.

[15] C.-T. Huang, J.-R. Huang, C.-F. Wu, C.-W. Wu, and T.-Y. Chang, "A programmable BIST core for embedded DRAM," IEEE Des. Test Comput., vol. 16, no. 1, pp. 59–70, Jan./Mar. 1999.

[16] J. -F. Li, R.-S. Tzeng, and C.-W. Wu, "Diagnostic data compression techniques for embedded memories with built-in self-test," J. Electron. Test.: Theory Appl., vol. 18, pp. 515–527, 2002.

[17] S. Hamdioui and J. E. Q. D. Reyes, "New data-background sequences and their industrial evaluation for word-oriented random-access memories," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 24, no. 6, pp. 892–904, Jun. 2005.

[18] S. Hamdioui, Z. Al-Ars, and A. J. van de Goor, "Opens and delay faults in CMOS RAM address decoders," IEEE Trans. Comput., vol. 55, no. 12, pp. 1630–1639, Dec. 2006.

[19] W.-L. Wang, K.-J. Lee, and J.-F. Wang, "An on-chip march pattern generator for testing embedded memory cores," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 9, no. 5, pp. 730–735, Oct. 2001.

[20] D.-C. Huang and W.-B. Jone, "A parallel built-in self-diagnostic method for embedded memory arrays," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 21, no. 4, pp. 449–465, Apr. 2002.

[21] B. H. Fang and N. Nicolici, "Power-constrained embedded memory BIST architecture," in Proc. 18th IEEE Int. Symp. Defect Fault Tolerance VLSI Syst. (DFT), 2003, pp. 451