Online Testing of Word-oriented RAMs by an Accumulator-based Compaction Scheme in Symmetric Transparent Built-In Self Test (BIST)

Sharvani Yedulapuram[#1], Chakradhar Adupa[*2]

[#]*Electronics and Communication Engineering*
Jayamukhi Institute of Technology and Sciences
Narsampet, Andhra Pradesh, India

***Abstract*—The paper presents a new approach to symmetric transparent Built-in self test (BIST) for word-oriented RAMs. Transparent built-in self test (BIST) schemes for RAM modules assure the preservation of the memory contents during periodic testing. The proposed concept allows to Skip the signature prediction phase required in traditional transparent BIST schemes, achieving considerable reduction in test time. In this paper the utilization of accumulator modules for output data compaction in symmetric transparent BIST for RAMs is proposed. It has been simulated & synthesized with Xilinx Spartan 3E based xc3s500e FPGA device. It is shown that in this way the hardware overhead, the complexity of the controller and the aliasing probability are considerably reduced.**

***Keywords*—Online testing, random access memories (RAMs), self testing, march algorithms.**

I. INTRODUCTION

The ASIC industry, driven by ever-increasing demands for miniaturization, higher reliability, and greater speeds, continuously introduce new product innovations to the micro electronics market place. In large ASICs the fault density is highest in RAM areas, and in many cases the RAM blocks which need to be tested are not directly accessible. Thus in order to build reliable and competitive products, it is essential that the testing strategy provide high fault coverage, without increasing a significant area overhead degrading the performance.

Memory is an important part of a computer system. Defects in memory arrays are generally due to shorts and opens in memory cells, address decoder and read/write logic. Memories are more likely to fail than random logic and therefore three classes of memory tests have been proposed to detect the memory faults. Application of test sequences to embedded memories using off-chip testers results in a high test time and test cost due to the large size of embedded memories. To overcome this problem, the computed test sequences are generated on-chip using a memory Built-In Self Test (BIST) unit. A number of theoretical and practical built-in self test (BIST) approaches, which have been proposed in the past, offer the basis to overcome the problem in present-day systems-on-a-chip [1], [3], [4]. RAMs are typically discerned into bit- and word-organized. For the testing of embedded RAMs, March algorithms outperform competitive schemes, since they result in simple, yet effective, testing scenarios. A March algorithm comprises a series of March elements that perform a predetermined sequence of operations (read and/or write) in every cell (for the case of bit-organized RAMs) or word (for the case of word-organized RAMs).

For deterministic memory BIST March tests have been widely accepted, because they combine a high fault coverage with a test time of order n, where n denotes the size of the memory [2], [13]. Furthermore, classical march tests can easily be extended to transparent tests which leave the Memory contents unchanged and therefore are especially suitable for periodic maintenance testing [3], [5]. The basic idea is to complement the memory contents an even number of times such that the memory state can be resumed while fault coverage can be maintained. A new signature prediction concept is used to verify the correctness of test responses, though extra test time must be invested to the signature prediction phase. However, 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.

To eliminate the signature prediction phase in [3] and [5], a symmetric transparent BIST method is proposed by Yarmolik [6], [7]. In symmetric transparent BIST, the signature prediction phase is skipped and the March series is modified in such a way that the final signature is equal to the all-zero state, irrespectively of the RAM initial contents. For response compaction of bit organized RAMs, in [6] a single-input shift register [SISR] was utilized, whose characteristic polynomial toggles between a primitive polynomial and its reciprocal one during the different march elements of the March series. For the case of word-organized RAMs, it was proven in [7] that a

multiple-input shift register (MISR) whose characteristic polynomial is altered in a similar fashion could serve as response compactor for symmetric transparent BIST, resulting in a predetermined (all-zero) state. The concept of symmetric transparent BIST is analyzed and exemplified in section III.

It is widely accepted by the test community that utilization of modules that typically exist in the circuit, e.g., accumulators [8] or arithmetic logic units (ALUs) [9], for BIST test pattern generation and/or response verification possesses advantages, such as lower hardware overhead and elimination of the need for multiplexers in the circuit path.

In this paper, we propose the use of accumulator-based compaction in symmetric transparent RAM BIST (ASTRA) by using March X algorithm. In modules that contain accumulators, the output of the RAM is either directly driven to the accumulator inputs or can be driven using processor instructions. It is shown that the proposed scheme imposes lower hardware overhead and less complexity in the control circuitry than previously proposed schemes.

This paper is organized as follows, in section II, describes the fault model for word-oriented RAMs. In section III, a review of March algorithms. In section IV, the proposed accumulator-based compaction scheme for symmetric transparent BIST (ASTRA) is introduced and exemplified for the case of word-organized RAMs. Next, in section V, the proposed scheme is compared to previously proposed schemes for response compaction in symmetric transparent BIST. In section VI, describes about Simulation results. Finally, in section VII, we conclude the paper.

II. FAULT MODEL FOR WORD-ORIENTED RAMs

Several innovative test algorithms for Random Access Memories have been reported in the recent years. In this paper, March X algorithm is used as a basis of test method. With this algorithm, the test complexity is reduced as shown in the Table I, where n denotes the size of the memory. The fault models for word-oriented memories can be divided into the following classes [12]:

A. Single-cell faults

This can be included as the following faults:

1) *Stuck-at faults:* cell or line s-a-0 or s-a-1

2) *Transition faults:* cell fails to transit

3) *Data retention faults:* cell fails to retain its logic value after some specified time due to, e.g., leakage, resistor opens, or feedback path opens

B.Fault between memory cells

This class of faults consists of coupling faults. Coupling faults are of three types:

1) *Inversion coupling fault (CFin):* a transition in one cell (aggressor) inverts the content of another cell (victim).

TABLE I
COMPARISON OF TEST COMPLEXITIES FOR CONVENTIONAL TRANSPARENT AND SYMMETRIC TRANSPARENT MARCH TESTS

Algorithm	Transparent BIST			Symmetric Transparent BIST
	Time for Signature Prediction	Time for Transparent Test	Overall Time	Total Time
MATS+	2n	4n	6n	4n
March X	5n	9n	14n	10n
March Y	4n	14n	18n	16n
March C-	5n	9n	14n	10n
March A	4n	14n	18n	16n
March B	6n	16n	22n	18n

2) *Idempotent coupling fault (CFid):* a transition in one cell forces a fixed logic value into another cell.

3) State coupling fault (CFst): a cell/line is forced to a fixed state only if the coupling cell/line is in a given state (a.k.a. pattern sensitivity fault (PSF)).

March tests for bit-oriented memories can be converted to march tests for word-oriented memories by taking into account that in the bit-oriented memory tests, the 'Rd0', 'Rd1', 'Wr0' and 'Wr1' operations are applied to a single bit. Word-oriented memories contain more than one bit per word; i.e., B≥2, that B represents the number of bits per word and usually is a power of two [11]. Read operations read the B bits simultaneously and write operations write data into the B bits of memory. Many different data backgrounds are used for testing of word-oriented memories [12].

III. MARCH ALGORITHMS

A. Traditional March Algorithms

A March algorithm consists of n march elements, denoted by M_i, with $(0 \le i < n)$. Each march element comprises zero (or more) write operations, denoted by w_0/w_1, meaning that 0/1 is written to the RAM cell, and zero (or more) read operations denoted by r_0/r_1, meaning that 0/1 is expected to be read from the memory cell. For example, the March X algorithm [Fig. 1(a)] consists of four March elements denoted by M_0 to M_3. In Fig. 1, ⇑ denotes an increasing addressing order (which can be any arbitrary addressing order) and ⇓ denotes a decreasing addressing order (which is the inverse addressing order of ⇑).

B. 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 [5] proposed the concept of transparent BIST where the initial w_0 phase is bypassed, and a signature prediction phase is used instead. The signature prediction phase consists of read operations equivalent to all 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 March X 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 r_0, r_1, w_0, w_1, 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).
- r_a^c Read the contents of a word of the RAM, expecting to read the compliment of the initial contents of the RAM word.
- $(r_a)^c$ Read the contents of a word of the RAM expecting to read the initial word contents and feed the compliment value to the compactor.
- 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
- 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 shown in [5] that with 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 (M_0) are identical to the read elements of the testing phase (M_1-M_3), then if we store the result of the compaction of M_0 and compare it to the result of the compaction of M_1-M_3, 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%. In order to confront this problem, Yarmolik, introduced the concept of symmetric transparent BIST, which is explained below.

C. Symmetric Transparent BIST

In order to define a symmetric transparent algorithm, some notations will be introduced first, Let $d = (d_0, d_1, .., d_{n-1}) \in \{0,1\}^n$ be a data stream, then $d^* = (d_{n-1}, d_{n-2}, \ldots, d_1, d_0)$ denotes

M_0	$\Updownarrow (w_0)$;	$\Uparrow (r_a)$;	$\Uparrow ((r_a)^c)$;
M_1	$\Uparrow (r_0, w_1)$;	$\Uparrow (r_a, w_a^c)$;	$\Uparrow (r_a, w_a^c)$;
M_2	$\Downarrow (r_1, w_0)$;	$\Downarrow (r_a^c, w_a)$;	$\Downarrow (r_a^c, w_a)$;
M_3	$\Downarrow (r_0)$;	$\Downarrow (r_a)$;	$\Downarrow (r_a)$;
	(a)	(b)	(c)

Fig.1. March X algorithm: (a) Original version; (b) Transparent version; and (c) Symmetric version.

-the data stream with components in reverse order and $d^c = (d_0^c \ldots d_{n-1}^c)$ denotes the data stream with inverted components. For example, if d = (1011), d* = (1101) and d^c = (0100).

A data string $D \in \{0, 1\}^{2n}$ is called **symmetric**, if there exists a data string $d \in \{0, 1\}^n$ with D = (d, d*) or D = (d, d^{*c}). For example, D_1 = (1010 0101) and D_2 = (1010 1010) are symmetric data strings, since (0101) = (1010)* and (1010) = $(1010)^{*c}$. Furthermore, a transparent march test is called symmetric if it produces a symmetric test data string D.

In order to derive a symmetric transparent algorithm, the March series is modified in such a way that the expected output response is equal to a known value. Therefore, the signature prediction phase can be skipped and the time required for the test is reduced.

In order to achieve this, Yarmolik, [6] noticed that most of the March algorithms used for transparent BIST produce test data with a high degree of symmetry. For example, the read elements of the transparent March X algorithms [Fig. 1(b)], ignoring the signature prediction phase (and the write elements) are: $\Uparrow (r_a)$; $\Downarrow (r_a^c)$; $\Downarrow (r_a)$ It is easy to detect the approximate symmetry; furthermore, it is also easy to derive a symmetric sequence by adding an additional read element, resulting in the following sequence of read elements: $\Uparrow ((r_a)^c)$; $\Uparrow (r_a)$; $\Downarrow (r_a^c)$; $\Downarrow (r_a)$. For example, for bit-organized memory with five words whose initial contents are (11010), the result of the latter sequence is (00101 11010 00101 | 01011 10100 01011) which is easily shown to be symmetric with respect to the given definition.

Yarmolik, [6], [7] have shown that by exploiting the previously mentioned symmetry and by using linear structures as compactors for the outputs of the RAM, the final value of the compactor is equal to a known value, i.e., the all-zero value. For the case of bit-organized memories, SISRs were utilized, while for word-organized memories MISRs were exploited, In [6] and [7], it was proven that by toggling between a primitive polynomial and its reciprocal one during the $\Uparrow$ r and $\Downarrow$ r operations, the final signature is equal to the all-zero state. They even reported marginal increase in the fault coverage of the symmetric schemes compared to the

respective transparent ones with signature prediction. For example, in Fig. 1(c) the symmetric transparent version of the March X algorithm is presented.

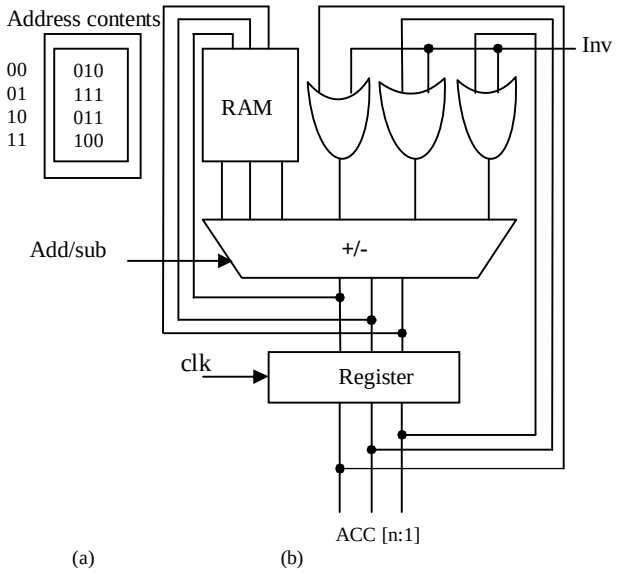

Fig. 2 Accumulator-based compaction in word organized memories.

IV. ACCUMULATOR-BASED COMPACTION OF THE RESPONSES IN SYMMETRIC TRANSPARENT BIST

The accumulator-based response compaction scheme proposed in this paper stems from the following two observations:

A. 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 r_a^c elements plus the number of $((r_a)^c)$ elements (without taking into account the addressing order, ⇑, ⇓ of the march element).

B. 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 [10]).

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 *M* 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.

TABLE II
OUTPUT DATA COMPACTION IN SYMMETRIC TRANSPARENT BIST: COMPARISON

Word-organized memories		
Scheme	Existing module	#Gates
[7]	Register	12×n
	MISR	6×n
Proposed	Accumulator	n

Therefore, for every vector **a** driven to the inputs of the accumulator, its compliment $\mathbf{a^c}$ is also driven to the inputs of the accumulator exactly once.

But

$$\mathbf{a} + \mathbf{a^c} = 2^n - 1 \qquad (1)$$

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

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

From (1) and (2), and taking in to account that the addition operation is permutative (observation 2), we have the proof:

For example, Let us consider the 4-word 3-bit RAM presented in Fig. 2(a), the outputs of the memory are driven to an n = 3-stage accumulator comprising a 1's compliment adder, Fig. 2(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 OR gates whose one input is driven by the inv signal in Fig. 2(b). Therefore, the inverse of the read vector appears at the outputs the adder/subtractor and applied to the RAM inputs.

V. COMPARISONS

In this section, we shall compare the proposed scheme with the one proposed in [7], with respect to the required hardware overhead and the test complexity. For the scheme proposed in [7], a MISR with double-shift (i.e. both left-to-right and right-to-left) capability is required; in case that a register is available, the transformation of *n* flip-flops into multiplexed input, two-way flip-flops is required; furthermore, *n* two-input XOR gates (to invert the values of the outputs of the RAM) and another *n* two-input XOR gates are required (in case A MISR is not available, in order to transform the register into a MISR). In case that a MISR exists, the transformation of *n* multiplexed input flip-flops into two-way multiplexed input flip-flops is required. The overhead is presented in Table II.

For the implementation of the proposed scheme, assuming the existence of an accumulator, *n* two-input OR gates are required at the inputs of the accumulator. Since the outputs of the RAM can be driven to the inputs of the accumulator by proper control of the data path module, no additional overhead is imposed.

From Table II, it is evident that the proposed scheme requires lower hardware overhead than the scheme proposed in [7] for the same purpose.

As compared to the symmetric transparent version of March C- algorithm (Yarmolik), the proposed algorithm is having less test complexity (Table 1) with high fault coverage.

VI. SIMULATION RESULTS

In this work we have designed a high speed accumulator based compaction scheme for online BIST of RAMs. The code is written in verilog language. Xilinx ISE 9.2i is the tool used to convert the verilog language to hardware blocks and implemented on Xilinx Spartan 3e based Xc 3s 500e FPGA device. The benefits associated with FPGA such as flexibility, shorter time to market and reconfigurability make them a very attractive choice for implementing the designs. In the Xilinx simulation environment, we create an ise file, then we specify the features of the device such as the device family, device, package, speed grade, synthesis tool, simulation tool, top level module type and the language. Then we add the source files i.e. the code and its test bench. Thus the ise file is created. Test bench with several test cases were setup to verify the expected results. After building the files we could able to see the design overview of the top level file, device utilization summary and performance summary. The figure 3 shows the RTL schematic of the top module and the figure 4 shows the result of our top module simulation, where the accumulator is holding the all-1 value. If the accumulator holds some other value then it results that there is a fault in the RAM module.

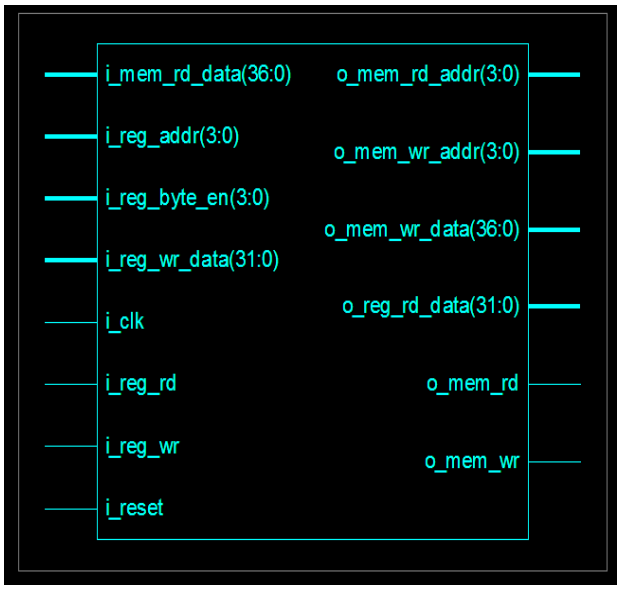

Fig.3. RTL Schematic of the top module

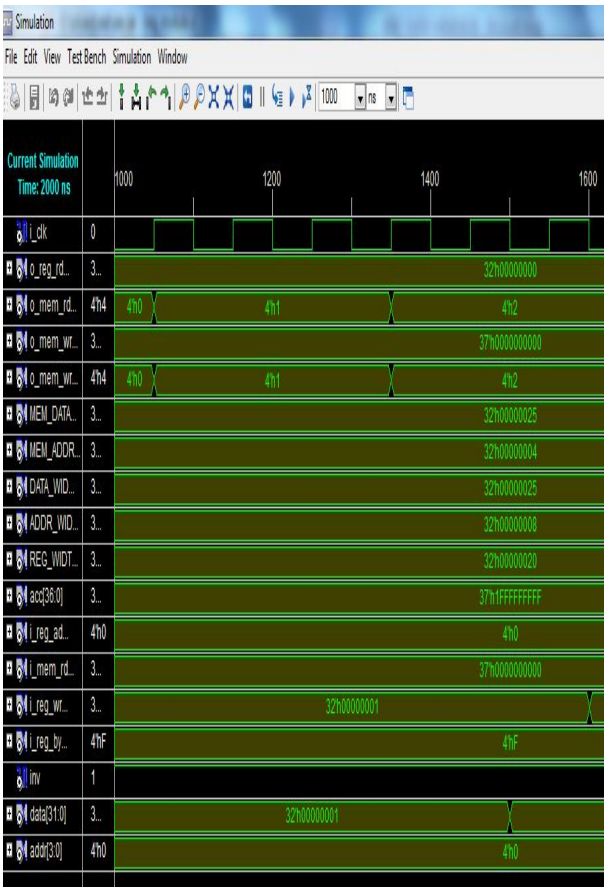

Fig.4. Result of our simulation

HDL Synthesis Report of the designed BIST is as follows:

Macro Statistics:

# Adders/Subtractors	: 2
37-bit adder carry in	: 1
5-bit addsub	: 1
# Registers	: 122
1-bit register	: 113
3-bit register	: 1
32-bit register	: 1
37-bit register	: 4
4-bit register	: 2
5-bit register	: 1
# Comparators	: 4
3-bit comparator greater	: 1
37-bit comparator equal	: 1
5-bit comparator greater	: 1
5-bit comparator less equal	: 1

Device utilization summary:

Selected Device: 3s500epq208-4

Number of Slices:	233	out of	3584	6%
Number of Slice Flip Flops:	279	out of	7168	3%
Number of 4 input LUTs:	411	out of	7168	5%
Number of IOs:	160			
Number of bonded IOBs:	160	out of	141	113%
Number of GCLKs:	1	out of	8	12%

VII. CONCLUSION

In this paper, we have proposed the utilization of accumulators for the compaction of the responses in ASTRA. Compared to traditional transparent BIST schemes this new approach significantly reduces the test time while preserving the benefits of previous approaches with respect to hardware overhead and fault coverage. Experimental studies even show an increase in fault coverage in many cases. Therefore, it may prove a viable solution for periodic testing of RAMs embedded into current VLSI chips.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their constructive comments that helped improve the quality of this work.

REFERENCES

[1] V. C. Alves, M. Nicolaidis, P. Lestrat, and B. Courtois: Built-in Self-Test *for Multi-Port RAMs; Proc. IEEE Int. Conf. on Computer-Aided* Design, ICCAD-91, November 1991, pp. 248-251.

[2] R. Dekker, F. Beenker, and L. Thijssen: A Realistic Fault Model and Test Algorithms for Static Random Access Memories; IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol 9, No. 6, June 1990, pp 567-572.

[3] M. Nicolaidis: Transparent BIST for RAMs; Proc. IEEE Int. Test Conf., Baltimore, MD, Oct. 1992, pp. 598-607.

[4] V. N. Yarmolik, S. Hellebrand, H.J. Wunderlich: Self Adjusting Output Data Compression: An Efficient BIST Technique for RAMs; Proc. Design and Test in Europe (DATE'98), Paris, February 1998, pp. 173-179.

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

[6] V. N. Yarmolik, S. Hellebrand, and H.J. Wunderlich, "Symmetric transparent BIST for RAMs," presented at the DATE, Munich, Germany, March.1999.

[7] 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.

[8] 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.

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

[10] 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.

[11] A.J, Van de Goor, I.B.S. Tlili, "March Test for Word-Oriented Memories", In Proc. Design Automation and Test in Europe, Paris, 1998, pp. 501-508.

[12] A.J, Van de Goor, I.B.S. Tlili, and S. Hamdioui, "Converting March Test for Bit-Oriented Memories into Tests for Word-Oriented Memories", IEEE Design and test, 1998, pp. 46-51.

[13] A. J. Van de Goor: Using March Tests to Test SRAMs; IEEE Design & Test of Computers, Vol. 10, No. 1, March 1993, pp. 8-14.