Testing of RAM's Through Symmetric Transparent Online BIST

[1]Jamanna Sreekanth Reddy, [2]K Bala Chandra, [3]T Lalith Kumar
[1]M.Tech Student, [2]Assistant Professor, [3]Associate professor
Department of Electronics and Communication Engineering
Sri venkateshwara institute of science and technology

***Abstract* - Transparent BIST schemes for RAM modules assure the preservation of the memory contents during periodic testing. Symmetric Transparent Built-in Self Test (BIST) schemes skip the signature prediction phase required in traditional transparent BIST, achieving considerable reduction in test time. Previous works on both offline MARCH-C testing scheme and online testing symmetric transparent BIST schemes require that a separate BIST module is utilized for each RAM under test. This approach, given the large number of memories available in current chips, increases the hardware overhead of the BIST circuitry. In this paper we propose a Symmetric transparent online BIST scheme that is used to test RAMs of different word widths; hence, more than one RAMs can be tested in a roving manner and here in this paper we are testing 5 RAMs of word lengths 3 bit to 7 bit . The hardware used for proposed scheme is smaller compared to the previously utilized proposed symmetric transparent schemes, for typical memory configurations.**

I. INTRODUCTION

Design for Testability (DFT) techniques are required in order to improve the quality and reduce the test cost of the digital circuit, while at the same time simplifying the test, debug and diagnose tasks. Built-in self-test (BIST) is a design for testability technique in which a portion of a circuit on a chip, board, or system is used to test the digital logic circuit itself. Memory Built-In Self-Test has become a standard industrial practice, since memory cores are making up a major part of the die area; it is forecasted that by 2Ol4 they will take up 94% of the die area. In addition, they are designed with minimal design rule tolerances, making them more susceptible to defects. Testing of RAM modules is performed both right after manufacturing and periodically in the field. During manufacturing testing, various kinds of tests are applied in order to ensure that the RAM operates normally. A march test comprises a series of march elements that perform a predetermined sequence of operations (read and/or write) in every word. Traditional march algorithms, start with an initial write-all-zero phase, where all the RAM cells are set to 'O' in order to ensure that the final signature in the output compactor is known.

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. Testing during normal operation, where the RAM normal operation is stalled (i.e. set out of normal operation), tested and then given back to operation, are applied to circuits where it is difficult and/or impractical to shut down the system since the contents of the RAM cannot be lost (e.g. space applications, wireless sensor network nodes, etc). In this kind of testing traditional march tests cannot be applied.

To deal with soft errors during system operation, adding standard online checking capabilities based on error detecting codes has been proposed [22]. Depending on the specific code, the detection of certain types of errors can be guaranteed. But, since error detection is only possible during read operations, the time between the occurrence of an error and its detection, referred to as error detection latency, may be very high. For some applications with high reliability requirements, e.g., in telecommunication switching, it is not acceptable to detect erroneous data only at the moment when the data are explicitly needed [23]. In contrast, errors should be detected as early as possible to allow for recovery before the data are requested by the system. Furthermore, error detecting codes have to increase the number of check bits to reduce the probability of masking multiple errors, which results in a high hardware overhead.

Transparent Built-In Self Test (BIST) was proposed by Nicolaidis [8]-[9] in order to confront these problems; in transparent BIST, the initial write-all-zero phase is skipped and a *signature prediction* phase is issued, during which a signature is captured and stored. In the sequel, a sequence of carefully selected read and write operations are performed, that leave the RAM contents equal to the initial ones; the final signature is compared to the one captured during the signature prediction phase and a decision is made as to whether a fault has occurred in the RAM or not. Transparent BIST schemes have been also proposed in [lO]- [l2].

One of the issues arising when transparent BIST is employed is that of the test data generator and response compactor. For traditional march algorithms the design of these two modules is trivial since known background patterns are applied (e.g. the all-O and all-l pattern), therefore a single signal is applied to the inputs of the data bus and two gates (one AND and one OR) are enough to verify the correct operation of the memory. In transparent BIST the need to capture the contents of the data contained in the memory at the beginning of the transparent test imposes the need to employ Multiple Input Shift Registers (MISR) structures, increasing the hardware overhead of the specific modules.

The idea of transparent BIST was further evolved by Yarmolik *et al* [l3], [l4] who proposed symmetric transparent BIST. In symmetric transparent BIST, the signature prediction phase is skipped and the march series is modified in such way that the

final signature is equal to the all-zero state, irrespective of the RAM initial contents. For response compaction of bit organized RAM's, in [l3] 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 RAM's it was proved that a Multiple-Input Shift Register (MISR) whose characteristic polynomial is altered in a similar fashion can serve as response compactor [l4]. The scheme requires the modification of existing registers (or SISRs / MISRs) in order to serve as response evaluators and requires appropriate control logic in order to toggle between the two different polynomials during the application of the march series and the modification is not needed in case of the proposed symmetric transparent online BIST as well as the traditional transparent BIST.

The idea of utilizing modules that typically exist in the circuit, e.g. accumulators [l5] or ALU's [l6], 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; furthermore, the modules are exercised, therefore faults. existing in them can be discovered [l7]. This idea is also behind the well-known concept of software- or processor- based BIST, where instructions of a processor are applied, using existing modules, to test the various modules of the chip [l8]-[l9].

In [2O], [2l] a Symmetric Transparent RAM BIST scheme was proposed, where the compaction and data generation module was implemented utilizing an ALU. In circuits that contain ALUs, the output of the RAM is either directly driven to the inputs of the ALU or can be driven using processor instructions. It was shown that the scheme [2l] imposes lower hardware overhead and less complexity in the control circuitry than previously proposed schemes.

On the other hand, as memory cores represent a significant portion of a multi-core chip's area (for example, Sun Microsystems' third-generation Ultras ARC multithreaded microprocessor has 94O memories, with a total of 27 million bits) although a BIST circuit's area cost is usually low, it increases as the number of small memories in a chip grows. Therefore, reducing the required BIST circuits becomes an issue for chips containing many memory cores. In order to test memories with the same word width in a transparent way, one can use the same transparent BIST module, that have been proposed e.g. in [l3], [l4], [2l] in a roving manner. However, the proposed schemes can not be utilized to test memories having different word widths. Hence, this would require separate BIST modules and, therefore, increased hardware overhead.

In this work we present a Symmetric Transparent Online BIST scheme for Arrays of Word-Organized RAMs (Star Wars). The proposed scheme utilizes an ALU in order to generate the test patterns and compress the responses of the memory module; the word width of the memory can be smaller than the number of stages of the ALU. The proposed scheme can be utilized to test transparently an array of memories in a roving manner, provided that the largest width of the memory does not exceed the number of stages of the ALU. Hence, multiple, non-identical memories can be tested in a pipeline way and the area cost is drastically reduced.

The paper is organized as follows. In Section 2 a review of the previous work on march algorithms (traditional, transparent and symmetric transparent) is given. In Section 5 the proposed Symmetric Transparent Online BIST for Arrays of Word-Organized RAMs (STArWoRs) is introduced and exemplified. In Section 4 the proposed scheme is compared to previously proposed schemes for response compaction and test data generation in symmetric transparent BIST. Finally, in Section 4 we conclude the paper. The conclusion gives the difference for the previous work and the proposed system i.e.,symentric online BIST algorithm.

II. PREVIOUS WORK

A march algorithm consists of n march elements ,denoted by Mp with 0 < i< n.each march element comprises 0 and1 when you want to test the data in the RAM will change.(or more) write operations, denoted by wO / wl meaning that O / l is written to the RAM cell, and zero (or more) read operations denoted by rO / rl, meaning that O / l is expected to be read from the memory cell. For example, the C- algorithm (Figure l(a)) consists of six march elements denoted by MO to M5 [5]. In Figure l, ž 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).

M_O.	□ (w_0);	□ (r_a); □ $((r_a))$; (r_a); $((r_a))$; $(_{ra})$;	□ $((r_a))$;
M_1.	□ (r_O, w_l);	□ (r_a, w_a);	□ (r_a, w_a);
M_2.	□ (r_l, w_O);	^{c}, w);	□ (r , w_a);
M_3.	(r_O, w_l);	(r_a, w_a^c);	(r_a, w_a^c);
M4.	(r_l, w_O);	(r_a^c, w_a);	(r_a^c, w_a);
M_5.	(r);	(r);	(r);
	(a)	(b)	(c)

Figure 1: C- march algorithm (a) original version (b) transparent version (c) symmetric transparent version

Traditional march algorithms erase the memory contents prior to testing; therefore, they do not serve as good platforms for periodic BIST. Nikolaidis [8] proposed the concept of transparent BIST where the initial wO phase is bypassed, and a "signature prediction" phase is used instead. The signature prediction phase consists of read operations 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 Figure l(b). The notation for the transparent versions of the algorithms differs from the one used in traditional march algorithms. Instead of rO, rl, wO,wl, the notations ra, rac, wa, wac and (ra)c are utilized, clarified in Table l.

Table 1: Notations for symmetric transparent BIST

Notation	Meaning
ra	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 complement 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 complement 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 definition, the data driven to the compactor with the (ra)c operation are identical to the data driven by the rac. The operation these contents are inverted prior to entering the compactor.

It has been shown [8], [9] that in transparent BIST the content of the memory at the end of the test is identical to that before the test. Also, since the read elements of the 'signature prediction' phase (MO) are identical to the read elements of the testing phase (Ml-M5), if we store the result of the compaction of MO and compare it to the result of the compaction of Ml-M5, then we can detect faults that occur, due to the write operations of the march algorithm. However, traditional transparent BIST has the disadvantage that the signature prediction phase adds up to the total testing time with a percentage of (more than) 3O%.

In order to confront this problem, Yarmolik *et al* introduced the concept of symmetric transparent BIST [l3][, [l4]; they first defined the concept of symmetric data stream as follows. Let $d = (d_O, d_l, \ldots, d_{n-l}) \in \{O,l\}^n$ be a data stream; then $d^* = (d_{n-l}, d_{n-2}, \ldots, d_l, d_O)$ denote the data stream with components in reverse order and $d^c = (d^c, \ldots d^c)$ denote the data stream with complemented components. For example, if d=(lOll), d* = (llOl) and d^c= (OlOO). A data string $D \in \{O, l\}^{2n}$ is called *symmetric*, if there exists a data string $d \in \{O, l\}^n$ with D = (d, d^*) or D = (d, d^{*c}). For example, D_l = (lOlO OlOl) and D_2 = (lOlO lOlO) are symmetric data strings, since (OlOl) = $(lOlO)^*$ and (lOlO) = $(lOlO)^{*c}$. Furthermore, a *transparent march test is called symmetric* if it produces a symmetric test data string D. Inorder to derive a symmetric transparent algorithm, the march series is modified in such 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, the authors of [l4] 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 C- march algorithms (Figure l(b)), ignoring the 'signature prediction' phase and the write elements, are: ž (ra); ž (rac); (ra); (rac); (ra);. In order to derive a symmetric sequence they add an additional read element, resulting in the following sequence of read elements: ž ((ra)c); ž (ra); ž (rac); (ra);

The authors of [l3], [l4] exploited the above-mentioned symmetry to test word-organized memories as follows. They utilized Multiple-Input Shift Registers (MISR's) and by toggling between a primitive polynomial and its reciprocal one during the ž r and r operations, the final signature is equal to the all-zero state. In Figure l(c) the symmetric transparent version of the C- algorithm is presented.

The accumulator-based symmetric transparent BIST solution proposed in [2O], [2l] utilizes an accumulator with an one's complement adder and stems from the observationsthat (a) if the march algorithm is symmetric, 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 and (b) 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 [l3].

III. PROPOSED SCHEME

In order to present the proposed scheme that can be utilized for more than one memory with varying number of word widths, in the sequel we shall present how we can perform transparent BIST when the number of stages of the ALU is larger than the memory word width.

3.1 Symmetric transparent BIST for memories with smaller word width than the number of the ALU stages

For the description of the proposed scheme, we will denote with n the number of stages of the ALU that can perform one's complement addition and with k the number of bits of the RAM word (hence k<n). The purpose of the proposed scheme is to assure that the contents of the register will be equal to a specific value (i.e. 'll…l') at the end of the test. In order to assure this, the (n-k) high-order inputs of the ALU are appropriately driven by the all-l or all-O value.

It should be noted that, if the march algorithm is symmetric, then the inputs driven to the response verifier and data generator during consecutive march elements, are complementary. In order to expand this for the case where the width of the memory is smaller than the number of the ALU stages, we add a series of (n-k) 'l's, i.e. a $\{l^{n-k}\}$ pattern at the high-order inputs of exactly the half elements added to the ALU. This stems from the fact that if $a \in 2^k$, since

$$\boldsymbol{a} + \boldsymbol{a^c} = 2^k\text{-}l,$$

then we also have that

$$n \in l$$

$$(\,|2^{i}\,)+ a + a^{c} = 2^{n}\text{-}1$$

Using the above observation, we can extend the scheme proposed in [2l] in order to handle the case where the ALU has more stages than the RAM word width. More precisely, we can stuff the high-order bits with a signal that has the value 'l' during half the cycles and 'O' during the other half. The idea is exemplified in the Figure 2.

The operation is different for read and write operations where for the read cycle and write cycle we consider the register contents as A and the RAM contents as B. The operation performed for the contents of algorithm are also different for one another, first consider a read cycle in read cycle the inverter is 0 so that the register contents come as it is because for a OR gate if one input is 0 then the output is the second input so this moves into ALU and the other data comes from RAM here for (Ra)' the subtraction operation is performed where as for remaining it performs addition operation. For a written cycle the inverter is 1 so that the output from OR gates is all one's and the data what has to be complemented that will be complemented when it subtracts from one's, thus it performs this symmetric online BIST architecture

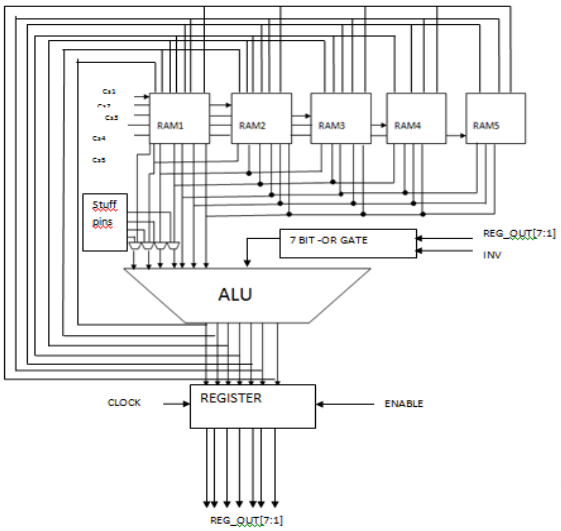

Figure 2: The proposed scheme for transparent testing of RAM with 3-bit words using a 5-stage ALU

In Figure 2 we present the situation where a memory with 3- bit words is transparently tested with a 5-stage ALU. The 5-3=2 high-order inputs of the ALU are driven by the signal *Stuff*.

In Table 2 we illustrate the operation of the module of Figure 2 for the case where the RAM has 4 words. We assume that the initial contents of the RAM words are {0l0, lll, Oll, l00}. In Table 2, in the first column we present the march element; in the second column we present the operation performed on the RAM. The number in the parentheses denotes the address of the accessed RAM word. In the third column we present the contents of the address; in the following columns we present the value of the *inv* signal, the contents that are written to the address (for the write operations), the value of the add/sub signal, the input to the ALU, comprising the additional inputs (Stuff signal) and the output of the RAM (for the read operations) and the contents of the register in every cycle. The value of the signal Stuff is 'll' exactly half the times of the test. We can see that the final value of the register for the case of the fault free RAM is the all-l value.

3.2 Transparent on line BIST for an array of RAM modules

The discussion of the previous subsection can be exploited in order to transparently test more than one memory modules, having different word widths in a roving manner. We exemplify the idea in Figure 3, for the case where three RAM modules, having words with 3, 4, and 5 bits each, respectively, are to be transparently tested on line in a roving manner using a 5-stage ALU. The RAM to be tested is enabled through the *csl*, *cs2* or *cs3* chip select signal respectively. When RAMl is tested, the inputs of the ALU are driven by the outputs of the RAMl; when RAM2 is tested, the high-order input of the ALU is driven by the *Stuff2* signal; when RAM3 is tested, the two high-order inputs are driven by the signals *Stuffl* and *Stuff2*, in a fashion similar to the one presented in Figure 2. Similarly we have a stuff1 to stuff4 to test the ram modules from RAM1 to RAM5.The RAM1 is a 7 bit so it does not uses any stuff pins RAM2 uses a single stuff pin since it is a 6 bit similarly RAM3, RAM4, RAM5 uses 3,4,5 stuff pins Because they are 5,4,3bits

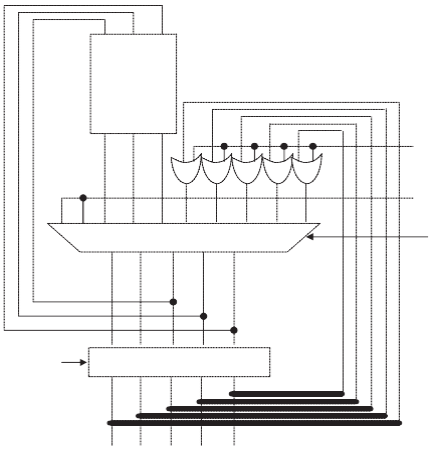

Figure 3: Transparent testing of three RAM modules with different word widths

respectively. Only one RAM is selected at a time for testing using a corresponding enable pin of that RAM. To select a different RAM we have to enable only after the testing of running RAM. The testing time is reduced as compared with the previous work on transparent online BIST algorithm.

IV. CONCLUSIONS

In this work we have presented a scheme for the testing of RAM modules using the symmetric transparent principle. The proposed scheme tests a RAM utilizing an ALU module whose number of stages can be larger than the RAM word width and naturally evolves into a scheme that can be used to test an array of RAM modules where the largest RAM word width does not exceed the number of stages of ALU Comparative results with schemes that have been proposed previously for symmetric transparent BIST of RAM modules [14], [21], indicate that the decrease of the hardware overhead, for the typical case where an 8-, a 16- and a 32-bit width word RAM modules exist in the circuit, and only a 32-stage BIST module exists, is between 84%- 85%.

V. ACKNOWLEDGMENTS

This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Archimedes III. Investing in knowledge society through the European Social Fund.

REFERENCES

[1] X. Du, N. Mukherjee, W-T Cheng, S. M. Reddy, "A Field-Programmable Memory BIST Architecture Supporting Algorithms and Multiple Nested Loops", Proc. of the Asian Test Symposium, paper 45.3, 2006.
[2] A. J. van de Goor, "Using March Tests to Test SRAMs", IEEE Design and Test of Computers, 1993.
[3] R. Aitken, et. al, "A Modular Wrapper Enabling High Speed BIST and Repair for Small Wide Memories", Proc. of Int. Test Conference, pp. 997-1005, 2004.
[4] X. Du, N. Mukherjee, W.T Cheng and S.M Reddy, "Full- speed field-programmable memory BIST architecture", Proc. of Int. Test Conference, pp. 1173-1182, 2005.
[5] I. Voyiatzis, "An ALU based BIST scheme for Word- organized RAMs", IEEE Transactions on Computers, vol. 57, no. 58, May 2008, pp. 577-590.
[6] V. N. Yarmolik, S. Hellebrand, H.-J. Wunderlich,"Symmetric Transparent BIST for RAMs", in Date 1999, Munich, Germany, March 9-12, 1999.
[7] V.N. Yarmolik, I.V. Bykov, S. Hellebrand, H.-J. Wunderlich, "Transparent Word-Oriented Memory BIST based on Symmetric March Algorithms", in European Dependable Computing Conference, 1999.
[8] I. Voyiatzis, "Test Vector Embedding into Accumulator- generated sequences: A Linear-Time Solution", IEEE Transactions on Computers, April 2005.
[9] A. Stroele, "BIST Patter Generators Using Addition and Subtraction Operations", Journal of Electronic Testing: Theory and Applications, vol. 11, pp. 69-80, 1997.
[10] R. Dorsch, H.-J. Wunderlich, "Accumulator-Based Deterministic BIST", International Test Conference, pp. 412-421, 1998.
[11] J. Zhou and H.-J. Wunderlich, "Software-Based Self Test of Processors under Power Constraints", Proc. Design, Automation and Test in Europe Conf. (DATE 06), 2006, pp. 430-435.
[12] V. N. Yarmolik, S. Hellebrand, H.-J. Wunderlich, "Symmetric Transparent BIST for RAMs", in Date 1999, Munich, Germany, March 9-12, 1999.
[13] V.N. Yarmolik, I.V. Bykov, S. Hellebrand, H.-J. Wunderlich, "Transparent Word-Oriented Memory BIST based on Symmetric March Algorithms", in European Dependable Computing Conference, 1999.
[14] I. Voyiatzis, "Test Vector Embedding into Accumulator- generated sequences: A Linear-Time Solution", IEEE Transactions on Computers, April 2005.