DETECTION OF PATTERN SENSITIVE MEMORY FAULTS

I. Mrozek

Bialystok University of Technology, Faculty of Computer Science, Wiejska str. 45a, 15-531 Bialystok, POLAND, imrozek@ii.pb.bialystok.pl

ABSTRACT

This paper develops the new solution for memory testing based on transparent memory tests in terms of pattern sensitive faults detection. Previous research has outlined that the only march tests can be in use now to test modern memory chips. Their transparent versions are very efficient for the simple fault diagnoses. The solution has been proposed in this paper dealing with the extension of known algorithms for the case of pattern sensitive faults. Using the proposed technique it is possible to detect pattern sensitive memory faults with a very high probability. Experimental investigations show the efficiency of the technique for simple faults, as well as for pattern sensitive faults.

1. INTRODUCTION

As the complexity of digital systems continues to increase, testing is consuming a larger proportion of the overall development costs. To overcome the shortcomings of the traditional external testing of the complex digital systems, different self-test techniques have been proposed in many articles. Built-In Self-Test (BIST) has become a major design for testability vehicle in the last fifteen years and has been employed at all levels of digital designs. The density of memory chips as the major part of digital systems has been increasing at a fast pace. Memory chips having 1GB and higher are on the horizon. More and more digital designers comprise numerous embedded arrays. Modern computer systems typically contain a variety of embedded memories like cashes, branch prediction tables or priority queues for instruction execution. This trend is expected to continue in the foreseeable future.

The increasing of integration density of these memories has made the problem of bit-oriented as well as word-oriented random access memories (RAM) testing dramatically complex because algorithms of complexity of $O(N^2)$ and $O(N^{3/2})$, where N is the number of cells in the memory, are no longer acceptable, At the same time the high fault probability makes it necessary to carry out testing more frequently. For efficient periodic testing, the memory contents have to be recovered at the end of the test, what is very important for a wide range of embedded memories. For deterministic memory BIST march tests have been widely accepted, because they combine a high fault coverage with a test complexity $O(N)$. Furthermore , classical march tests can be easily extended to transparent tests which preserve the memory contents unchanged and therefore are especially suitable for periodic maintenance testing. Transparent versions of march tests are very efficient for the simple faults diagnoses. This paper presents march tests efficiency for static, active and passive pattern sensitive faults. Over that a new test technique has been presented for pattern sensitive faults.

2. TYPES OF MEMORY FAULTS

There are many kind of memory faults . The most important of them are presented below[3].

2.1. Simple Faults

Simple faults are faults which are not linked; they can be classified as *address decoder faults*(*AFs*), and as *memory cell array faults* (*MCAFs*) consisting of single cell faults and faults between memory cells. An *address decoder fault* (*AF*) can only be present in the address decoder. Memory cell array faults we can divide into next way:

- single faults
  - *Stuck-at* (*SAF*). A permanent stuck-at 0 or stuck-at 1 fault that may occur in any memory cell
  - *Stuck-open fault* (*SOF*). An SOF in a memory cell means that the cell cannot be accessed
  - *Transition fault* (*TF*). A cell fails to undergo $0 \rightarrow 1$ transition and/or $1 \rightarrow 0$ transition.
  - *Data retention fault* (*DRF*). A cell fails to retain its logic value after some period of time.
- faults between memory cells

Coupling fault (CF) involves two cells (aggressor cell and victim cell). A write operation that affects a 0->1 or a 1->0 state transition in cell j (aggressor), changes the state of another memory cell i (victim) ($i \neq j$) from s to not s, independently of the contents of the other cell. We can classifies the coupling faults as following:

- indempotent (or noninverting) coupling faults, that is: A positive (or negative) transition in a cell cj forces another cell c_i to a certain value 0 or 1;
- inverting coupling faults, that is: A positive (or negative) transition in cell c_j inverts the state of a cell c_i, irrespective of the value of this state.
- disturb faults (CFds) that is: A fault whereby the victim cell is disturbed due to r*y* or a w*y* operation ($y \in \{0,1\}$) applied to aggressor cell.

Pattern Sensitive Faults [1] - The contents of a cell, or ability to change the contents, is influenced by the contents of all other cells in the memory. These contents consists of a pattern of 0s and 1s, or changes in these contents. The PSF can be considered the most general case of the k-coupling fault, namely the case whereby k=n (whereby n represents all cells in the memory).

2.2. Linked Faults

A *linked faults* involves two or more simple faults. A CF is linked with another CF when both CFs have the same victim cell (this may cause fault masking). When a TF is linked with a CF, and TF is in the aggressor cell, the CF may not be sentizable. When the TF is in the victim cell, TF may be masked.

2.3Neighborhood Pattern Sensitive Faults

A **Pattern Sensitive Fault** (PSF) is defined as follows [1]: The contents of a cell, or ability to change the contents, is influenced by the contents of all other cells in the memory. These contents consists of a pattern of 0s and 1s, or changes in these contents. The PSF can be considered the most general case of the *k-coupling* fault, namely the case whereby $k=n$ (whereby n represents all cells in the memory).

The **neighborhood** is the total number of cells involved in a particular fault model. The **base cell** is the cell under test. The neighborhood with the base cell excluded is called the **deleted neighborhood**.

The PSF model allows the deleted neighborhood to take on any position in the memory array. When the deleted neighborhood is allowed to take on only a single position (such that it surrounds the base cell), one speaks about a **Neighborhood Pattern Sensitive Fault** (NPSF). The NPSF model is therefore a subset of the PF model. Three types of NPSFs can be distinguished:

- **Active NPSF** (ANPSF) also called a **Dynamic NPSF**: the base cell changes its contents due to a change in the deleted neighborhood pattern. This changes consists of a transition in one deleted neighborhood cell, while the remaining deleted neighborhood cells and the base cell contain a certain patter.
- **Passive NPSF** (PNPSF): the content of the base cell cannot be changed (it cannot make a transition) due to a certain neighborhood pattern.
- **Static NPSF** (SNPSF): the content of a base cell is forced to a certain state due to a certain deleted neighborhood pattern.

3. MARCH TESTS

March test consists of family of test which all have the same structure; they have proven to be superior in terms of test time and simplicity. This section introduces the notation used to describe march tests and shows examples of different march tests.

A *march test* consists of a sequence of march elements; a *march element* consists of a sequence of operations which are all applied to a given cell, before proceeding to the next cell. The way one proceeds to the next cell is determined by the address order which can be an increasing address order (increasing addresses from cell 0 to n-1), denoted by the "⇑", or the decreasing address order, denoted by the "⇓" symbol. The "⇑" address order has to be exact inverse of the "⇓" address order (Van de Goor 1991). For some march elements the address order can be chosen arbitrary, this will be indicated by the "↕" symbol. An operation applied to a cell, can be 'w0' (write '0') , a w1, a "r0" (read 0), or a "r1" operation.

A complete march test is delimited by the "{...}" bracket pair; while a march element is delimited by the "(...)" bracket pair. The following march test

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

is the MATS+ test (Nair, 1979). It consists of three march elements , M_0, M_1 and M_2. M_2 performs a r1 operation followed by a w0 operation on a cell, after which these two operations are applied to the next cell.

4. Transparent Tests

The limited reliability of large high-density memory chips requires periodic field-testing without affecting the memory contents. This is also important for applications requiring to verify the integrity of a system before delivering a critical service; for applications using concurrent checking, e.g., duplication, error detecting codes, where periodic testing is needed to avoid long fault latency that could enable accumulation of faults; and for fault tolerant systems based on recovery (fault detection must be guaranteed between check points) and eventually using reconfiguration by means of spare units. Preservation of the RAM contents after testing can be achieved by a transparent BIST scheme.

Following the notation defined in previous section the original MATS+ algorithm can be written as $\{\Updownarrow(w(0); \Uparrow(r0, w1); \Downarrow(r1,w0)\}$ It can be transformed into transparent test by deleting the initialization phase and replacing read and write operation with fixed "0" or "1" values by read an write operations with the appropriate values [2]. For the transparent version $\{\Uparrow(ra, w\bar{a}); \Downarrow(r\bar{a}, wa);\}$ 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. This reference signature, however, depends on the memory contents and must be computed in 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 tests. For the transparent MATS+ algorithm it is described by $\{\Uparrow(ra); \Downarrow(r\bar{a})\}$ and requires an extra time of 2n. The time for the 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.

5. Using March Tests to Detect Pattern Sensitive Faults.

Pattern sensitive faults involve many memory cells. Therefore it is very difficult to detect pattern sensitive faults [1]. In this section, results of experiments concerning detection of pattern sensitive faults using march tests with the new proposed test technique has been presented. In contrast to the traditional test techniques in proposed technique we don't have to calculate the initial characteristic which we have to calculate in traditional transparent techniques.

As has been mentioned, there exist three kinds of PSF [1]. As these faults involve many memory cells simultaneously they are very difficult to detect. Let us consider an example of a memory with eight cells where three of them are involved in Static PSF: 011 – base cell, 001 and 101 deleted cells. The base cell is forced to the value '1' when the both deleted cells contain value '1'. Let us test this memory using non-transparent March LA test for the first time and the transparent one for the second time.

a) testing memory using non-transparent test

The non-transparent March LA test can be presented in the following way:

$\{\Updownarrow(w0); \Uparrow(r0,w1,w0,w1,r1); \Uparrow(r1,w0,w1,w0,r0);$
P1 *P2*
P3
$\Downarrow(r0, w1,w0, w1,r1); \Downarrow(r1,w0,w1,w0,r0); \Downarrow(r0)$
P4
P5 *P6*

	000	001	010	011	100	101	110	111
Phase 1	0	0	0	0	0	0	0	0
Phase 2	1	1	1	1	1	1	1	1
Phase 3	0	0	0	0	0	0	0	0
Phase 4	1	1	1	1	1	1	1	1
Phase 5	0	0	0	0	0	0	0	0

The fault activates in the 2-nd and the 4-th phases but the test doesn't detect it

b) testing memory using transparent test

The transparent March LA test can be presented in the following way:

$\{\Uparrow(ra, w\bar{a}, wa, w\bar{a}, r\bar{a}); \Uparrow(r\bar{a}, wa, w\bar{a}, wa, ra);$
P2
P3
$\Downarrow(ra, w\bar{a},wa, w\bar{a}, r\bar{a}); \Downarrow(r\bar{a},wa,w\bar{a},wa,ra); \Downarrow(ra,$
P4
P5 *P6*

	000	001	010	011	100	101	110	111	S ref
START	1	0	1	0	1	0	1	0	000
Phase 2	0	1	0	1	0	1	0	1	000
Phase 3	1	0	1	0	1	0	1	0	000
Phase 4	0	1	0	1	0	1	0	1	000
Phase 5	1	0	1	0	1	0	1	0	000

The fault activates in the 2-nd and the 5-th phases but the test doesn't detect it

Let us analyze the same fault as well as the same test but with a different initial memory state.

	000	001	010	011	100	101	110	111	S ref
START	1	0	0	1	1	1	0	1	101
Phase 2	0	1	1	1	0	0	1	1	001
Phase 3	1	0	1	0	1	0	1	0	
Phase 4	0	1	0	1	0	1	0	1	
Phase 5	1	0	1	0	1	0	1	0	

It is visible that with this initial memory state the existing fault is detected.

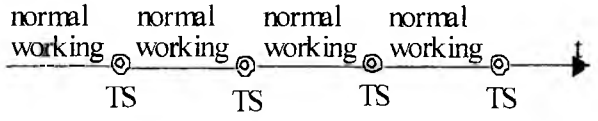
It is visible from the examples mentioned above that there exists some faults which can't be detected by non-transparent tests. Using the transparent tests the success of the test depends on the memory contents in the moment of the test initiation. Assuming both random changes of the memory contents and frequently enough memory testing we can come to a conclude that traditional March algorithms (in the transparent version) can be successfully used in the detection of pattern sensitive faults. Tabl. 1 presents the probability of the detection of pattern sensitive memory faults by the transparent version of some well known march tests

Table 1. Pattern sensitive faults coverage (in %)

TEST	Probability of detect of pattern sensitive faults		
	PASSIVE	STATIC	ACTIVE
MATS+	24,71	38,47	25,78
MATS++	24,71	36,13	23,05
MARCH C-	48,93	65,62	67,58
MARCH A	24,71	76,46	74,90
MARCH B	26,56	99,99	75,29
MARCH LA	48,93	65,92	75,68

We can see that the probability of detection of pattern sensitive faults is rather low. To increase this probability we have to test the memory more frequently, but if the time period between test sessions is too short, the change of the memory contents may be too insignificant to allow the detection existing pattern sensitive faults. On the other hand if the time period between test sessions is too long, it is dangerous, the detection of fault will be too late.

The new idea allowing the increase of probability of the detection of pattern sensitive faults is presented below. Using this technique we can detect pattern sensitive faults in the first test session after its appearance. During one test session we run the march test many times. Between tests we generate a random bit's pattern. According to the generated pattern we change memory contents. After each test in the test session we come back to the original memory contents using the same bit's pattern. We can write following scheme of memory testing:



where

- t – time
- normal working – normal working of memory
- TS - test session

According to the standard memory testing technique during one test session we run test only once. According to the proposed technique we run march test many times during one test session. So one test session we can write in the following way:

```
March_test()
for (i=0 i<k;i++)
{
        b=random_bit's_pattern()
        memory = memory XOR b
        March_test()
        memory=memory XOR b
}
```

where k – number of march tests during one test session

To make possible this idea we have to use:

- the transparent test;
- memory contents independent test technique.

Every march test we can transform to the transparent version[6].As for a memory contents independent test technique – it is a technique based on modulo-2 address characteristic proposed. As shown in Figure 1, the modulo-2 address characteristic compresses the memory contents to a characteristic *C* obtained as the bitwise modulo-2 sum of all addresses pointing to "1"[2]. As opposed to the traditional test technique, based on modulo-2 address characteristic, in the proposed technique we don't have to calculate the initial characteristic which we have to calculate in traditional technique based on modulo-2 address characteristic [2]. The proposed technique is fully independent of memory size and contents. We calculate the characteristic in each phase of march tests. It is obvious that if there aren't

any faults in the tested memory, the characteristics in all phases of the test should be identical

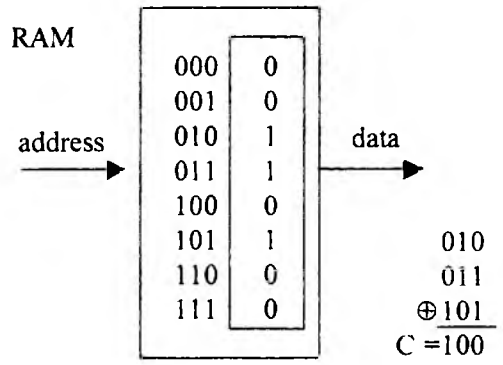


Figure 1. Modulo-2 address characteristic for bit-oriented RAMs.

Therefore after ending each phase of the test we compare calculated characteristic with characteristic from the previous phase (except for the first phase of course). In the case of March LA test:

$$\{\Uparrow(ra, w\bar{a}, wa, w\bar{a}, r\bar{a});\ \Uparrow(r\bar{a}, wa, w\bar{a}, wa, ra);$$
$$P2$$
$$P3$$
$$\Downarrow(ra, w\bar{a}, wa, w\bar{a}, r\bar{a});\Downarrow(r\bar{a}, wa, w\bar{a}, wa, ra);\ \Downarrow(ra)\}$$
$$P4$$
$$P5 \qquad\qquad P6$$

we compare, after the ending of phase *P3*, - characteristic from phase *P3* with characteristic from phase *P2*, after ending of phase *P4* - characteristic from phase *P4* with characteristic with phase *P3*, after ending of phase *P5* - characteristic from phase *P5* with characteristic from phase *P4* and after ending of the last phase - characteristic from phase *P6* with characteristic from phase *P5*.

The test session time using the proposed technique depends on the probability of the pattern sensitive faults coverage we want. We can write that:

$$T=k*t+\text{extra\_time}$$

where

- T – test session time,
- k – number of test running during one test session
- t – time of one test using during test session
- extra_time – time for generating the new bit's pattern and changing memory contents between tests during one test session.

Tabl. 2 shows the resulting test lengths for some commonly used transparent march tests [7].

Table 2. Test time of comonly used transparent march tests.

Algorithm	Signature prediction	TEST	Total time
MATS +	2n	4n	6n
March C--	5n	9n	14n
March A	4n	14n	18n
March B	6n	16n	22n
March X	3n	5n	8n
March LA	9n	21n	30n

We have to remember that according to the proposed idea, we don't need to calculate the initial value of the signature before running the test. Fig. 2 shows pattern sensitive fault coverage (in %) depending on the number of march test in the test session for March LA test

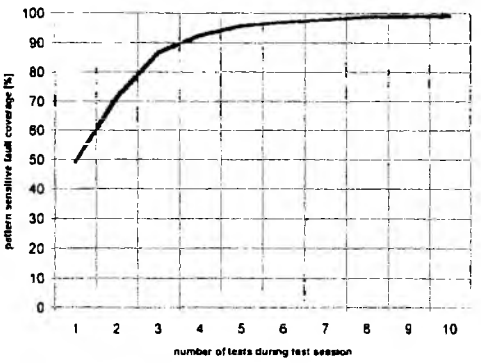


Figure 2: Simulation results (passive pattern sensitive fault coverage in %) for a 32Kbit memory

The following tables show simulations of results of passive, active and static pattern sensitive faults coverage for the proposed technique, using different march tests. Simulation was done for 32Kbit memory and many well known march tests. For each kind of march test the number of tests during one test session was changed from zero to ten. From the results we can see we should run about eight march tests during one test session to ensure the high pattern sensitive faults coverage.

Table 3. Passive pattern sensitive fault coverage in % for proposed technique

Test	number of tests during one test session				
	2	4	6	8	10
MATS +	56,25	77,63	84,96	91,41	95,02
MATS ++	56,25	76,54	86.17	90,20	96,01
MARCH C-	85,64	96,58	98,73	99,12	99,22
MARCH A	56,25	7426	85,70	92,20	94,03
MARCH B	58,60	74,41	88,09	91,50	94,53
MARCH LA	71,29	92,48	96,97	98,93	99,20

Table 4. Active pattern sensitive fault coverage in % for proposed technique

Test	number of tests during one test session				
	2	4	6	8	10
MATS +	48,63	62,30	62,69	63,76	68,35
MATS ++	48,92	60,25	64,55	67,48	65,00
MARCH C-	94,92	98,82	99,51:	99,99	99,99
MARCH A	97,94	99,70:	99,85	99,94	99,99
MARCH B	97,75	99,51	99,70	99,91	99,97
MARCH LA	98,73	99,71	99,91	99,98	99,99

8. CONCLUSIONS

This paper presents the comparison of the behaviour of transparent tests in terms of pattern sensitive faults detection. It is proposed that the new technique allows the detection of pattern sensitive faults with a very high probability. According to this technique during one test session we run march tests many times. From the experimental results we can see the very high probability of the detection of pattern sensitive faults, which we get when we use March LA or March C- tests.

Table 5. Static pattern sensitive fault coverage in % for proposed technique

Test	number of tests during one test session				
	2	4	6	8	10
MATS +	75,10	87,01	94,43	96,09	97,95
MATS ++	72,85	87,99	93,94	96,78	98,05
MARCH C-	95,11	97,96	99,98	99,99	99,99
MARCH A	97,16	99,71	99,80	99,99	99,99
MARCH B	99,99	99,99	99,99	99,99	99,99
MARCH LA	95,90	99,31	99,90	99,96	99,99

REFERENCES

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

[2]. V. N. Yarmolik, S. Hellebrand, H.J. Wunderlich: Self-Adjusting Output Data Compression: An Efficient BIST Technique for RAMs; Proceedings *Design, Automation an Test in Europe Conference 1998* (DATE 98), Paris, 1998, pp. 173-179

[3]. A.J. van de Goor, G.N. Gaydadjiev, V.N. Yarmolik, V.G. Mikitjuk March LA: test for All linked Memory Faults. In Proc. *14-th IEEE VLSI Test Symposium*, pp 272-280

[4]. Van de Goor, A.J., Gaydadjiev, G.N, Yarmolik, V.N., Mikitjuk V.G. (1996) March LR: A test for Realistic Linked Faults. In Proc. *14-th IEEE VLSI Test Symposium*, pp 272-280

[5]. V.N.Yarmolik, S. Hellebrand, H.J. Wunderlich, Symmetric Transparent BIST for RAMs, Proc. *of Designe and Tst in Europe* (DATA'99), Munich, March, 1999, pp. 702-707.

[6]. I. Mrozek, V.N. Yarmolik Symulacja destrukcyjnych i niedestrukcyjnych testów pamięci, zbór referatów z VI Warsztatów Naukowych PTSK Symulacja w badaniach i rozwoju pod redakcją L. Bobrowskiego i R. Bogacza - Białystok, *Białowieża – sierpień* 1999r, s. 385-396

[7]. A. J. Van de Goor: Testing Semiconductor Memories, Theory and Practise; *Chichester John Wiley & Sons*, 1991.

[8]. 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

[9]. M. Nicolaidis "An Efficent Built-In Self-Test for Functional Test of Embedded RAMs", Proc. 15th *Int'l Symp. Fault Tolerant Computing*, Ann Arbor, Mich, June 1985

[10]. A. J. Van de Goor: Using March tests to Test SRAMs, IEEE *Design & Test of Computers*, March 1993, pp. 8-14

[11]. G. N. Gaydadjiev,, A. J. Van de Goor, An analysis of Linked faults Technical report No 1-68340-44(1995)08, 1995 Delf University of Technology

[12]. B. Cockburn, Z.F.Sat Synthesised Transparent BIST for Detecting Scrambled Patter-Sensitive Faults in RAMs, Proc. *International Test Conference*, Washington D.C. October, 1995, pp. 23-32

[13]. Nair R.,Thatte S.M., Abraham J.A. Efficient Algorithm for Testing Simiconductor Random-Access Memories. IEEE *Trans. on Computers*, 1978, C-27, No 6, pp.572-576.

[14]. Papachristou C.A., Saghal N.B. An Improved Method for Detecting Functional Faults in Random Access Memories. IEEE Trans. on Computers, 1985, 2, pp.110-116.