March Test Algorithm for 3-Coupling Faults in Random Access Memories

CAŞCAVAL PETRU, ONEA ALEXANDRU
Dept. of Computer Science, Dept. of Automatic Control
"Gh. Asachi" Technical University of Iaşi
Bd. D.Mangeron, nr.53A, 6600, Iaşi
ROMANIA

Abstract: - A new efficient march test algorithm for detecting 3-coupling faults in Random Access Memories is given in this paper. To reduce the length of the test only the 3-coupling faults between physically adjacent memory cells have been considered. The proposed test algorithm needs *34N* operations. Simulation results with regard to the coupling fault coverage of the march tests, obtained based on a fault injection mechanism, are also presented in this paper. This work improves the results presented in [1] where the same problem is dealt with and a march test with *38N* operations is given.

Key-Words: RAM, memory testing, functional faults, coupling faults, march test, fault coverage

1 Introduction

Rapid developments in semiconductor technology resulted in continuing growth of larger and denser random access memories (RAM) on a single chip. More time is required to test memories because of their increasing size. On the other hand, as a result of the increased cell density the nature of the failure mode becomes more complex and subtle [2].

Memory test procedures are constrained by two conflicting requirements:

a) to detect a wide variety of complex faults;

b) to reduce the number of memory operations in order to allow the testing of large memory size to be carried out in an acceptable period of time.

Efficient test algorithms for detecting stuck-at faults and 2-coupling faults have been proposed, see for example Refs. [2-5]. All of these are march algorithms with a reduced number of operations.

For 3-coupling faults, a memory test that requires $N+36Nlog_2N$ operations is given by Nair, Thatte and Abraham [3]. Papachristou and Sahgal proposed in [4] a new algorithm with the same ability to detect 3-coupling faults but with only $37N+24Nlog_2N$ operations (*PS(B*) in this article). Unfortunately, for the memory chips currently available, these tests take a long time to perform. For example, assuming a cycle time of 100 ns, *PS(B)* takes about 4 min to test a 4Mb memory chip and 1h 14 min to test a 64-Mb memory chip. This time is not acceptable in many cases, such as the on-line testing. Both memory tests, are lengthy because the authors have assumed that the three coupled cells can be anywhere in the memory. This paper proposes a new march test for 3-coupling faults with an acceptable compromise between the fault detection ability and the length of the test. Thus only the 3-coupling faults that affect physically adjacent memory cells are considered.

This work improves the results presented in [1] where the same problem was treated.

2 Memory Fault Model

The fault models used in this paper have been formalised by Nair, Thatte and Abraham [3] and refined later by Papachristou and Sahgal [4]. At the same time some definitions and fault models have been draw from Suk and Reddy [5]. We have also adopted notations given by van de Goor [2].

This paper is focused on very difficult to detect faults such as 2-coupling and 3-coupling faults. For a group of coupled cells *transition* and/or *state coupling faults* may exist.

1) Transition coupling faults

a) *2-Coupling Faults* A write operation that affects a $0\rightarrow1$ or a $1\rightarrow0$ transition into cell *j* changes the state of another memory cell *i*, independently of the contents of the other cells. This does not necessarily imply that a state transition in cell *i* changes the contents of cell *j* [3]. Cell *i* is called the *coupled cell* and cell *j* is called the *coupling cell*. We can say that cell *j* has an *active influence* on cell *i*.

b) *3-Coupling Faults*. For a set of three coupled cells, a transition in one cell causes the state of another cell in the set to change from *0* to *1* or from

1 to *0*, when the third remaining cell has a fixed state.

2) State coupling faults

A state-coupling fault occurs when one or more cells in a group of v-coupled cells ($v \geq 2$) fail to undergo a $0 \rightarrow 1$ or a $1 \rightarrow 0$ transition when the complement of the contents of the memory cell is written into the cell [2]. This type of fault depends on the states of the remaining $v-1$ cells in the group. In this case, we can say that the remaining $v-1$ cells in the group have a *passive influence* on the coupled cell. When cell i is a coupled cell the fault is called *i-state coupling fault.*

In memory, one or more groups of coupled cells may exist. When the pairs of groups of coupled cells are disjoint the model is called *restricted coupling faults.* As in Refs. [3,4] we considered only this restricted model.

The interacting coupling faults are also accepted in this model [5]. Informally, the significance of two interacting faults is that their combined effects may cancel each other out.

In this paper only the physically adjacent cells have been considered. Two cells are physically adjacent if they have a border or even a corner in common. Fig. 1 shows all the four distinct patterns (P_1, P_2, P_3, P_4) for a group $S=(i,j,k)$ of three adjacent cells.

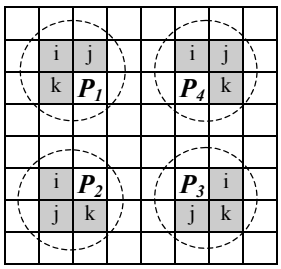

Fig. 1 Patterns for three physically adjacent cells

3 Preliminaries

Definition 1. For every memory cell in RAM three possible states can be considered [5]:

- *Internal state* is the actual content of the memory cell.
- *Apparent state* is the result of a read operation of a memory cell.
- *Expected state* is the expected content of a cell after one or more write operations.

Definition 2. Faults are *detected* if and only if one or more differences between the expected states and the apparent states of the cells occur during the test [5].

Definition 3. A forced transition is defined in [3] as one that is initiated by the testing algorithm by writing into a cell (of course, this may cause transitions in other cells because of coupling).

The following notations to describe operations on RAM's are used in this paper:

- R the read operation on a cell;
- W_x the operation of writing x into a cell, $x \in \{0,1\}$;
- W_c the operation of writing the complement of the previous apparent or expected state of a cell;
- $\uparrow i$ the operation of writing *1* into cell i when the previous apparent or expected state of cell i was *0*;
- $\downarrow i$ the operation of writing *0* into cell i when the previous apparent or expected state of cell i was *1*.

Consider S a group of v cells. To describe a failed transition in S we use the notation given by van de Goor [2]. Thus, a vector F with $v+1$ elements shows the conditions to sensitise the fault (the initial state of group S and the forced transition) and the result of fault sensitising (the change of state of the coupled cell specified in context, in the format: *good value/faulty value*). For example, for a group of cells $S=(i,j,k)$:

- if j is a coupled cell, vector $F=<0,\uparrow,0;1/0>$ describes a *state coupling fault* in which the transition $\uparrow j$ has no effect when cell i and k are in the state *0*.
- if k is a coupled cell, vector $F=<\uparrow,\phi,0;0/1>$ describes a *transition coupling fault* in which the forced transition $\uparrow i$ changes the state of cell k from *0* to *1*.

Definition 4. In this paper a fault is called *simple fault* if only one vector F is necessary for description the fault behaviour. If at least two vectors are necessary for a complete description the fault is called a *complex fault.*

Any complex fault can be modelled as a set of distinct simple faults simultaneously present in the memory. An interacting fault is a complex fault which comprises two or more (even number) simple faults with contrary influence on the same cell [1],[7]. Examples of complex fault:

- Take an $i \rightarrow j$ coupling fault in which the transition $\uparrow i$ changes the state of cell j from 0 to 1 and the transition $\downarrow i$ from 1 to 0. This complex interacting fault can be modelled by two vectors: $F_1=<\uparrow,0,\phi;0/1>$ and $F_2=<\downarrow,1,\phi;1/0>$.
- Take a linked coupling fault with cell j a coupled cell. Both transition $\uparrow i$ and $\uparrow k$ change the state of cell j from 0 to 1. This fault can be modelled by $F_1=<\uparrow,0,\phi;0/1>$ and $F_2=<\phi,0,\uparrow;0/1>$. This is not an interacting fault.

To detect a fault in a memory under testing two conditions are needed:

- to activate (to sensitise) the fault by a proper forced transition;
- to observe the fault by reading the changed value of the cell affected by the fault.

Remark 1. Generally, complex faults are easier to detect than simple faults because there are more input vectors for which complex faults are activated. Thus, if a test procedure detects any simple fault, moreover, it detects all complex non-interacting faults.

Assertion 1. The next three conditions are necessary and sufficient for a test to detect any simple fault in a group of coupled cells:
Condition 1. For a group of cells *S* the test must force all the possible cell transitions.
Condition 2. After a forced transition into a cell the test must read the cell to check if the state has changed before to force another transition into the cell.
Condition 3. Every cell must be read first before a forced transition, in order to check if the state has been changed by a transition in a coupled cell.

Remark 2. If for a group of cells *S* all the possible transitions are forced during the test (*condition 1* is satisfied) then all the possible faults that affect the cells in *S* are activated. *Conditions 2* and *3* are needed to observe the memory faults.

Definition 5. A march element *(M)* is a finite sequence of operations applied to every cell in the memory in either one of two orders, increasing address order from address zero (⇑) or decreasing address order from N-1 (⇓) [2]. Symbol ⇕ is used when the address order does not matter.

Definition 6. A march test is a finite sequence with *n* march elements $T = <M_1; M_2; \ ... \ ; M_{n-1}; M_n>$.

The march test may also comprise one or more sequences to initialise the memory.

4 March Tests Currently Used

The best-known march test algorithms currently used to detect coupling faults are presented bellow.
1) The march algorithm (*March C*) with *10N* operations presented in [2].
2) The march algorithm (*March B*) with *16N* operations given by Suk and Reddy [5].
3) The march algorithm (*March G*) with *24N* operations given by van de Goor [2].
4) The march test with *30N* operations given by Nair, Thatte and Abraham [3] (*NTA(A)* in this paper).
5) The march test with *37N* operations given by Papachristou and Sahgal [4] (*PS(A)* in this paper).

Experimental results regarding to the 2-coupling and 3-coupling fault coverage of these march tests are presented in table 1 and table 2, respectively. In order to simulate a permanent fault into the memory under test we have used a software fault injection mechanism based on *TRAP* microprocessor interrupt.

For every possible transition in a group of coupled cells *S* we have checked two kind of faults:

- *state coupling faults* - the forced transition does not change the state of the addressed cell;
- *transition coupling faults* - the forced transition changes the state of the addressed cell but, at the same time, it changes the state of another coupled cell in group *S*.

Table 1 Simple 2-coupling fault coverage (%)

March C (10N)	*March B (17N)*	*March G (24N)*	*NTA(A) (30N)*	*PS(A) (37N)*
81.25	81.25	100	100	100

For 3-coupling faults four groups of cells have been considered, one for each pattern P_i, $i \in \{1,2,3,4\}$ (see Fig.1). For each group, all possible simple 3-coupling faults were injected: 24 *state coupling faults* and 48 *transition coupling faults*. In total, for all groups of cells, we have checked 288 simple faults.

Table 2 Simple 3-coupling fault coverage (%)

March C (10N)	*March B (17N)*	*March G (24N)*	*NTA(A) (30N)*	*PS(A) (37N)*
41.67	47.22	62.50	63.89	63.89

All these march tests have good fault coverage for *2-coupling faults* but, as we can see in table 2, have low fault coverage for *3-coupling faults*. For example, the best of them, *NTA(A)* and *PS(A)*, cover completely 2-coupling faults but only 63.89% of simple 3-coupling faults.

A new efficient march test with *38N* operations able to detect all simple 3-coupling faults is given in [1], where the same memory fault model has been considered. In section 5 another march test to cover restricted 3-coupling faults which affect physically adjacent cells is presented. Comparing with the test presented in [1] this new march test needs only *34N* operations and is more adequate for a built-in self-testing implementation [8].

5 A New March Test Algorithm

In this section a new march test (*MT)* with *34N* operations is presented. This test procedure uses six different patterns (data background) for memory initialisation.

$MT = <I_1;\Uparrow(RW_cRW_c);\Updownarrow(R);\ I_2;\Uparrow(RW_cRW_c);\Updownarrow(R);$
$I_3;\Uparrow(RW_cRW_c);\Updownarrow(R);\ I_4;\Uparrow(RW_cRW_c);\Updownarrow(R);$
$I_5;\Uparrow(RW_cRW_c);\Updownarrow(R);\ I_6;\Uparrow(RW_cRW_c);\Updownarrow(R)>$

where I_1, I_2, I_3, I_4, I_5 and I_6 are sequences for memory initialisation (Fig.2):

- I_1 and I_3 initialises any cell with *0* and *1*, respectively (solid data background);
- I_2 initialises the odd columns with *0* and the even columns with *1*, and I_4 vice versa (column-stripe data background);
- I_5 initialises the odd rows with *0* and the even rows with *1*, and I_6 vice versa (row-stripe data background).

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	**0**	**0**	0	0	0	0	0	0	**0**	**0**	0	0
0	0	**0**	0	0	0	0	0	0	0	0	**0**	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	**0**	0	0	0	0	0	0	0	0	**0**	0	0
0	0	**0**	**0**	0	0	0	0	0	0	**0**	**0**	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

I_1

1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	**1**	**1**	1	1	1	1	1	1	**1**	**1**	1	1
1	1	**1**	1	1	1	1	1	1	1	1	**1**	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	**1**	1	1	1	1	1	1	1	1	**1**	1	1
1	1	**1**	**1**	1	1	1	1	1	1	**1**	**1**	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

I_3

0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	**1**	**0**	1	**0**	**1**	0	1	**0**	**1**	0	**1**	**0**	1
0	**1**	0	1	**0**	1	0	1	0	**1**	0	1	**0**	1
0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	**1**	0	1	**0**	1	0	1	0	**1**	0	1	**0**	1
0	**1**	**0**	1	**0**	**1**	0	1	**0**	**1**	0	**1**	**0**	1
0	1	0	1	0	1	0	1	0	1	0	1	0	1

I_2

1	0	1	0	1	0	1	0	1	0	1	0	1	0
1	**0**	**1**	0	**1**	**0**	1	0	**1**	**0**	1	**0**	**1**	0
1	**0**	1	0	**1**	0	1	0	1	**0**	1	0	**1**	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0
1	**0**	1	0	**1**	0	1	0	1	**0**	1	0	**1**	0
1	**0**	**1**	0	**1**	**0**	1	0	**1**	**0**	1	**0**	**1**	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0

I_4

0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	**1**	**1**	1	1	1	1	1	1	1	1	**1**	**1**	1
0	**0**	0	0	**0**	**0**	0	0	**0**	**0**	0	0	**0**	0
1	1	1	1	**1**	**1**	1	1	1	**1**	1	1	1	1
0	**0**	0	0	0	0	0	0	0	0	0	0	**0**	0
1	**1**	**1**	1	**1**	1	1	1	1	**1**	1	**1**	**1**	1
0	0	0	0	**0**	**0**	0	0	**0**	**0**	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1

I_5

1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	**0**	**0**	0	0	0	0	0	0	0	0	**0**	**0**	0
1	**1**	1	1	**1**	**1**	1	1	**1**	**1**	1	1	**1**	1
0	0	0	0	**0**	**0**	0	0	0	**0**	0	0	0	0
1	**1**	1	1	1	1	1	1	1	1	1	1	**1**	1
0	**0**	**0**	0	**0**	0	0	0	0	**0**	0	**0**	**0**	0
1	1	1	1	**1**	**1**	1	1	**1**	**1**	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0

I_6

Fig. 2 Data background used by *MT* and the possible initial states for all distinct patterns

Remark 3. Every march element RW_cRW_c applied to a memory cell leaves the cell into the same state because the cell is switched twice. Consequently, a march sequence leaves the memory into the same state if no fault has occurred.

Assertion 2.

The march test *MT* detects all simple 3-coupling faults which affect physically adjacent cells.

Demonstration

Consider a group of three physically adjacent cells *S=(i,j,k)* with addresses in increasing order ($i<j<k$).

We have to check that *MT* activates and observes any simple fault in group of cells *S*.

1. *MT* activates any fault in group of cells *S*

To activate any fault in group *S* the test program must force all possible transitions in this group (*assertion 1* in section 3). For a group of three cells *24* distinct transitions exist. Fig.3 shows the Eulerian graph of states for a group of three cells.

Every group of three physically adjacent cells

corresponds with one of the patterns P_1, P_2, P_3 or P_4 (see Fig.1). Consequently, we consider a group of cells for each pattern P_1, P_2, P_3 and P_4 and demonstrate that *MT* covers the graph of states.

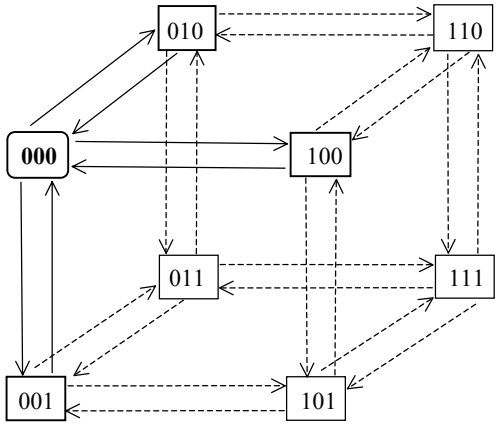

Fig. 3 The Eulerian graph of states for a group of cells $S=(i,j,k)$

We can see that two adjacent nodes (states) in the graph have only one bit changed and two non-adjacent nodes have at least two bits changed.

A march sequence in *MT* performs six different transitions in group of cells *S*. In the Eulerian graph three adjacent nodes are visited (going and coming back) and, finally, the group of cells returns to the initial state (before the sequence started). For example, the first sequence $I_1;\Uparrow(RW_cRW_c)$ performs the transitions marked with solid lines in the graph.

As highlighted in Fig.2 the initialisation sequences I_1, I_2, I_3, I_4, I_5 and I_6 ensure in a group of cells *S* the states presented in Table 3. Of course, these initial states are specific for each pattern.

Table 3 The initial states for a group *S*

	I_1	I_3	I_2 and I_4	I_5 and I_6
P_1	000	111	101, 010	110, 001
P_2	000	111	110, 001	011, 100
P_3	000	111	101, 010	100, 011
P_4	000	111	011, 100	001, 110

Table 4 The missing initial states for a group *S*

Pattern	States
P_1	011, 100
P_2	010, 101
P_3	001, 110
P_4	010, 101

For every group of cells only two initial states can not be obtained by the sequences I_1, I_2, I_3, I_4, I_5 and I_6. Table 4 gives the initial states which are missing.

For all patterns, the two missing initial states are complementary states as all bits are changed. In a geometrical interpretation (see Fig.3), the associated nodes in the graph of states are not in the same plane. In these conditions, it can easily check that the Eulerian graph of states is completely covered, in all the four cases. In other words, *MT* forces all possible transitions in every group of three physically adjacent cells and, consequently, is able to activate any fault in a group of coupled cells.

2. *MT* observes any simple fault activated in group *S*

MT repeats the pair of march sequences (RW_cRW_c) and (R) six times. Thus, for every memory cell a write operation (W_c) is preceded and succeeded by a read operation (R). Consequently, *condition 2* and *3* previously defined in section 3 (*assertion 1*) are satisfied. □

Because *MT* is able to detect any simple restricted 3-coupling fault it also covers the complex non-interacting 3-coupling faults (*remark 1* in section *3*). In addition, *MT* is also able to detect the interacting coupling faults, but this complex problem can not be treated here.

The simulation result confirms that *MT* detects all simple 3-coupling faults.

6 Conclusions

This paper presents a new efficient march test algorithm for coupling faults in random access memories. Only the coupling faults between physically adjacent cells have been considered.

MT needs *34N* operations and covers the 3-coupling fault model. We demonstrated analytically that *MT* is able to detect all simple 3-coupling faults using an Eulerian graph model.

Experimental results with regard to the coupling fault coverage of the best-known march tests are also given in this paper.

Taking into account the 3-coupling fault coverage, *MT* is better than all the march tests currently used.

Comparing with the non-march test *PS(B)*, also especially designed to cover 3-coupling faults, *MT* is considerable reduced. For example, *MT* takes 14.2s and *PS(B)* about 4 min to test a 4Mb memory chip if we assume a cycle time of 100 ns. In other words, for testing a 4Mb memory chip, *MT* is about 16.7 times faster than *PS(B)*.

This work improves the result presented in [1] where a march test algorithm with *38N* operations is

given.

MT is a homogeneous march test and comprises only two kind of march sequences, (RW_cRW_c) and (R). Consequently, *MT* is adequate for built-in self-testing implementation in embedded RAM [8].

References:

[1] Caşcaval P., Bennett S., Efficient March Test for 3-Coupling Faults in Random Access Memories, *Microprocessors and Microsystems*, Elsevier Science, Vol. 24, No. 10, 2001, pp.501-509.

[2] van de Goor A.J., Using March Tests to Test SRAMs, *IEEE Design and Test of Computers*, January-March, 1993, pp.8-14.

[3] Nair R., Thatte S., Abraham J., Efficient Algorithms for Testing Semiconductor Random-Access Memories, *IEEE Transactions on Computers*, Vol. C-27, No.6, 1978, pp. 572-576.

[4] Papachristou C., Sahgal N., An Improved Method for Detecting Functional Faults in Semiconductor Random Access Memories, *IEEE Transactions on Computers*, Vol. C-34, No.2, 1985, pp.110-116.

[5] Suk D.S., Reddy M., A March Test for Functional Faults in Semiconductor Random Access Memories, *IEEE Transactions on Computers*, vol. C-30, No.12, pp.982-985.

[6] Caşcaval P., Huţanu C., Silion R., Memory Fault Coverage Evaluation For March Tests, *Buletinul Institutului Politehnic Iaşi*, Tomul XLV (IL), Fasc.1-4, Automatică şi Calculatoare, 1999, pp.89-96.

[7] Caşcaval P., Interacting Coupling Faults in Random Access Memories, *Buletinul Institutului Politehnic Iaşi*, Tomul XLVI (L), Fasc.1-4, Automatică şi Calculatoare, 2000, pp.121-130.

[8] Caşcaval P., Onofrei V., Built-in Self-Testing for Coupling Faults in Random Access Memories, *Buletinul Institutului Politehnic Iaşi*, Tomul XLVI (L), Fasc.1-4, Automatică şi Calculatoare, 2000, pp.93-101.