ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **12**, Number 4, 2009, 440–454

A BIST Logic Design for $MarchS_3C$ Memory Test BIST Implementation

[1]Petru CAŞCAVAL, [2]Radu SILION, [3]Doina CAŞCAVAL

Gheorghe Asachi Technical University of Iaşi, Romania
E-mail: {[1]cascaval, [2]rsilion}cs.tuiasi.ro,
[3]cascaval@tex.tuiasi.ro

Abstract. A logic design for a possible built-in self-testing implementation of a march test able to detect all static simple three-cell coupling faults in $n \times 1$ random-access memories (RAMs) is presented. Single–array single bit and multiple–array single bit test architectures have been considered. The memory test ($MarchS_3C$ [1]) needs $66n$ operations and is able to detect all realistic simple (i.e. not linked) static three-cell coupling faults that have been shown to exist in real designs, namely: state coupling faults, transition coupling faults, write disturb coupling faults, read destructive coupling faults, deceptive read destructive coupling faults, and incorrect read coupling faults. To reduce the length of the test, only the coupling faults between physically adjacent memory cells have been considered. The test assumes that the storage cells are arranged in a rectangular grid and that the mapping from logical addresses to physical cell locations is known completely.

Key words: Memory testing, static faults, fault primitive, three-cell coupling faults, built-in self-testing.

1. Introduction

This article focuses on high performance memory testing having in view system on chip (SoC) dedicated to critical applications, with high reliability and safety requirements, in which the test confidence degree regarding to normal operation of the embedded memory must be very high. In recent years, embedded memories are the fastest growing segment of SoC. According to the 2001 International Technology Roadmap for Semiconductor [2], today's SoC are moving from logic dominant chips

to memory dominant chips, since future applications require a lot of memory. The same idea is reiterated in [3]. The memory shared on the chip is expected to be about 94% in 2014. For this reason, the importance of memory testing increases.

Rapid developments in semiconductor technology have resulted in continuing growth of larger and denser random-access memories on a single chip. With increasing densities, certain types of faults harder-to-detect such as three-cell or even four-cell coupling faults can not be ignored any more [4, 5]. Consequently, the test algorithms are constrained by two conflicting requirements: to cover a wide variety of memory faults harder-to-detect, and to reduce the number of memory operations in order to allow large memories to be tested in an acceptable period of time. In addition, more time is required to test memories because of their increasing size thus it is necessary to identify more efficient tests, with the ability to detect complex faults, tests that require test time on the order of n, where n denotes the number of locations of the memory. In this work, the class of faults harder-to-detect that includes coupling faults with three adjacent coupled cells has been considered.

BIST is a design-for-testability technique that places test functions physically on chip with the circuit under test. System designers use BIST for periodic testing. This requires incorporating a test process that guarantees the detection of all target faults within a fixed time. Designers also implement on-line BIST with the goals of large fault coverage and low fault latency. For critical or highly available systems, a comprehensive online-testing approach that covers all expected permanent, intermittent, and transient faults is essential [6].

Two kinds of testing methods exist: random testing and deterministic testing. Random testing is based on linear-feedback shift registers (LFSR) for pattern generation. An LSFR can also serve as a response monitor by counting the responses produced by the tests. After receiving a sequence of test responses, an LSFR response monitor forms a fault signature which is compared with a known good signature to determine whether a fault is present. Deterministic testing is especially suited to highly regular chips.

Since the RAM circuit has a regular structure, a deterministic testing is more adequate than a random one. Taking into account the number of simultaneously tested arrays and the number of simultaneously accessed bits within an array, Franklin and Saluja [6] classified all the RAM-BIST test architectures into one of the four test architectures: single-array single bit, single-array multiple bit, multiple-array single bit, and multiple-array multiple bit. In this work only single-array single bit (SASB) test architectures and multiple-array single bit test architectures (MASB) have been considered. SASB test architectures are those in which a single array of the RAM chip is tested at a time and a single bit of the tested array is accessed at a time. MASB test architectures can be used if a memory chip is organized as a number of independent arrays, allowing multiple arrays to be tested simultaneously. Ensuring that fault coverage is sufficiently high and the number of tests is sufficiently low are the main problems with a BIST implementation [5].

In this paper we focus on the model of all static simple three–cell coupling faults, as defined in [1]. This is the largest model of three-cell coupling that includes all the faults that have been shown to exist in real designs, namely: state coupling faults,

transition coupling faults, write disturb coupling faults, read destructive coupling faults, deceptive read destructive coupling faults, and incorrect read coupling faults [4, 7]. For a BIST-RAM logic design, the memory test $MarchS_3C$ that covers entirely this fault model has been considered [1].

The remainder of this paper is organized as follows. Section 2 presents a memory fault classification based on the concept of fault primitives. The fault primitives for the model of all static three-cell coupling are presented in Section 3. The memory test $MarchS_3C$ is presented briefly in Section 4. To compare $MarchS_3C$ with other published memory tests, simulation results are also presented in Section 5. A logic design for a possible implementation of this march test in a BIST-RAM device is discussed in Section 6. Final remarks regarding this work are presented in Section 7.

2. Primitives and a Memory Fault Classification

An operation sequence that results in a difference between the observed and the expected memory behaviour is called a *sensitizing operation sequence* (S). The observed memory behaviour that deviates from the expected one is called *faulty behaviour* (F). In order to specify a certain fault, one has to specify the S, together with the corresponding faulty behaviour F, and the read result (R) of S in case it is a read operation. The combination of S, F and R for a given memory failure is called a *Fault Primitive* (FP). The concept of FPs allows for establishing a complete framework of all memory faults. Some classifications of FPs can be made based on different and independent factors of S.

a) Depending on the number of simultaneous operations required in the S, FPs are classified into *single-port* and *multi-port* faults.

- *Single-ports faults:* These are FPs that require at the most one port in order to sensitize a fault. Note that single-port faults can be sensitized in single-port as well as in multi-port memories.

- *Multi-port faults:* These are FPs that can only sensitize a fault by performing two or more simultaneous operations via the different ports.

b) Depending on the number of sequential operations required in the S, FPs are classified into *static* and *dynamic* faults. Let #O be the number of different operations carried out sequentially in a S.

- *Static faults:* These are FPs which sensitize a fault by performing at the most one operation in the memory (#O=0 or #O=1);

- *Dynamic faults:* These are FPs that perform more than one operation sequentially in order to sensitize a fault (#O $>$ 1).

c) Depending on the way FPs manifest themselves, they can be divided into *simple faults* and *linked faults*.

- *Simple faults:* These are faults which cannot be influenced by another fault. That means that the behaviour of a simple fault cannot change the behaviour of another one; therefore masking cannot occur.
- *Linked faults:* These are faults that do influence the behaviour of each other. That means that the behaviour of a certain fault can change the behaviour of another one such that masking can occur. Note that linked faults consists of two ore more simple faults.

In this work, single-port, static, simple faults are considered. From here on, the term 'fault' refers to a single-port, static, simple fault.

The following notations are usually used to describe operations on RAMs :

- $r0$ ($r1$) denotes a read 0 (1) operation from a cell;
- r denotes a read operation from a cell when the expected value is not specified;
- $w0(w1)$ denotes a write 0 (1) operation into a cell;
- $0w0$ ($1w1$) denotes a write 0 (1) operation to a cell which contains a 0(1) − a non-transition write operation when the logical value of the cell does not change;
- w denotes a non-transition write operation into a cell when the logical value of the cell is not specified;
- $0w1$ ($1w0$) denotes an up (down) transition write operation;
- w_c denotes a transition write operation into a cell when the old logical value of the cell and its complement are not specified;
- $\uparrow$ ($\downarrow$) denotes an up (down) transition due to a certain sensitizing operation.

A FP is usually denoted as $< S/F/R >$ [6], where:

- S describes the value/operation sensitizing the fault, $S \in \{0,\ 1,\ r0,\ r1,\ 0w0,\ 1w1,\ 0w1,\ 1w0\}$;
- F describes the value of the faulty (*victim*) cell (v-cell), $F \in \{0,\ 1,\ \uparrow,\ \downarrow\}$;
- R describes the logical value which appears at the output of the memory if the sensitizing operation applied to the v-cell is a read operation, $R \in \{0,\ 1,\ -\}$. The symbol '$-$' in R means that the output data is not applicable; for example, if S=$0w0$, then not data will appear at the memory output, and therefore R is replaced by a '$-$'.

RAM faults can also be divided into *single-cell* and *multi-cell* faults. Single-cell faults consist of FPs involving a single cell, while multi-cell faults consists of FPs involving more than one cell. As concerns the multi-cell faults (also called *coupling faults*), we restrict our analysis to the class of three-cell FPs (i.e. three-cell coupling faults).

3. All Static Three-Cell Coupling Faults

Based on the notations previously defined, a three-cell FP is presented as $< S/F/R > = < S_{a1}\,; S_{a2}\,; S_v/F/R >_{a1,\,a2,\,v}$, where S_{a1}, S_{a2} and S_v are the sensitizing operation (or state) sequences performed on the $a1$-cell and $a2$-cell (*aggressor cells*), and on the v-cell (*victim cell*), respectively. The $a1$-cell and $a2$-cell are the cells to which the sensitizing operation (or state) should be applied in order to sensitize the fault, while the v-cell is the cell where the fault appears. Note that S_{a1}, $S_v \in \{0, 1, r0, r1, 0w0, 1w1, 0w1, 1w0\}$, whereas $S_{a2} \in \{0, 1\}$. As presented in Table 1, there are 72 FPs compiled into seven FFMs, as defined in [7], namely:

- state coupling faults (*CFst*);
- disturb coupling faults (*CFds*);
- transition coupling faults (*CFtr*);
- write destructive coupling faults (*CFwd*);
- read destructive coupling faults (*CFrd*);
- deceptive read destructive coupling faults (*CFdrd*);
- incorrect read coupling faults (*CFir*).

For example, three cells are said to have a disturb coupling fault if an operation (read, transition or non-transition write) performed on the $a1$-cell causes the v-cell to flip, when $a2$-cell is in a given state; the FP defined in the row 9 shows that a read 0 operation performed on $a1$-cell causes the v-cell to flip from 0 to 1, when the $a2$-cell is in 0 state. Note that, the number of fault primitives is equal to the number of arcs in the graph of states that describes the normal operations for three memory cells (8 states $\times$ 9 arcs for each state). For this reason, the model is called "all static three-cell coupling faults".

Many test algorithms dedicated to such models of three-cell coupling faults have been reported. Thus, a memory test that requires $n + 32n \log_2 n$ operations is given by Nair, Thatte, and Abraham (*Algorithm B*) [8]. A new test of length $n + 24n \log_2 n$ is proposed by Papachristou and Sahgal [9]. Two more efficient test algorithms, *S3CTEST* and *S3CTEST2*, are given by Cockburn [10]. These are tests of approximate length $5n \log_2 n + 22, 5n$ and $5n \log_2 n{+}5n\, [\log_2(1 + \log_2 n)] + 11n$, respectively. But, for the memory chips currently available, all these tests take a long time to perform because the authors have assumed that the coupled cells can be anywhere in the memory. For example, to test a 64 Mb memory chip assuming a cycle time of 60 ns, *S3CTEST* takes about 10 min 54s.

To reduce the length of the test, Caşcaval and Bennett have restricted the model to the more realistic coupling faults that affect only the physically adjacent memory cells [11]. Thus, for a set of three coupled cells $\{i, j, k\}$, six patterns denoted by P_1, P_2, P_3, P_4, P_5 and P_6 are accepted, as shown in Fig. 1.

Table 1. List of three-cell FPs

#	Fault primitives	FFM
1	< 0; 0; 0; / 1 / - >	*CFst*
2	< 0; 1; 0; / 1 / - >	
3	< 1; 0; 0; / 1 / - >	
4	< 1; 1; 0; / 1 / - >	
5	< 0; 0; 1; / 0 / - >	
6	< 0; 1; 1; / 0 / - >	
7	< 1; 0; 1; / 0 / - >	
8	< 1; 1; 1; / 0 / - >	
9	< *r*0; 0; 0; / ↑ / - >	*CFds*
10	< *r*0; 0; 1; / ↓ / - >	
11	< *r*0; 1; 0; / ↑ / - >	
12	< *r*0; 1; 1; / ↓ / - >	
13	< *r*1; 0; 0; / ↑ / - >	
14	< *r*1; 0; 1; / ↓ / - >	
15	< *r*1; 1; 0; / ↑ / - >	
16	< *r*1; 1; 1; / ↓ / - >	
17	< 0*w*0; 0; 0; / ↑ / - >	
18	< 0*w*0; 0; 1; / ↓ / - >	
19	< 0*w*0; 1; 0; / ↑ / - >	
20	< 0*w*0; 1; 1; / ↓ / - >	
21	< 1*w*1; 0; 0; / ↑ / - >	
22	< 1*w*1; 0; 1; / ↓ / - >	
23	< 1*w*1; 1; 0; / ↑ / - >	
24	< 1*w*1; 1; 1; / ↓ / - >	
25	< 0*w*1; 0; 0; / ↑ / - >	
26	< 0*w*1; 0; 1; / ↓ / - >	
27	< 0*w*1; 1; 0; / ↑ / - >	
28	< 0*w*1; 1; 1; / ↓ / - >	
29	< 1*w*0; 0; 0; / ↑ / - >	
30	< 1*w*0; 0; 1; / ↓ / - >	
31	< 1*w*0; 1; 0; / ↑ / - >	
32	< 1*w*0; 1; 1; / ↓ / - >	
33	< 0; 0; 0*w*1; / 0 / - >	*CFtr*
34	< 0; 1; 0*w*1; / 0 / - >	
35	< 1; 0; 0*w*1; / 0 / - >	
36	< 1; 1; 0*w*1; / 0 / - >	
37	< 0; 0; 1*w*0; / 1 / - >	*CFtr*
38	< 0; 1; 1*w*0; / 1 / - >	
39	< 1; 0; 1*w*0; / 1 / - >	
40	< 1; 1; 1*w*0; / 1 / - >	
41	< 0; 0; 0*w*0; / ↑ / - >	*CFwd*
42	< 0; 1; 0*w*0; / ↑ / - >	
43	< 1; 0; 0*w*0; / ↑ / - >	
44	< 1; 1; 0*w*0; / ↑ / - >	
45	< 0; 0; 1*w*1; / ↓ / - >	
46	< 0; 1; 1*w*1; / ↓ / - >	
47	< 1; 0; 1*w*1; / ↓ / - >	
48	< 1; 1; 1*w*1; / ↓ / - >	
49	< 0; 0; *r*0; / ↑ / 1 >	*CFrd*
50	< 0; 1; *r*0; / ↑ / 1 >	
51	< 1; 0; *r*0; / ↑ / 1 >	
52	< 1; 1; *r*0; / ↑ / 1 >	
53	< 0; 0; *r*1; / ↓ / 0 >	
54	< 0; 1; *r*1; / ↓ / 0 >	
55	< 1; 0; *r*1; / ↓ / 0 >	
56	< 1; 1; *r*1; / ↓ / 0 >	
57	< 0; 0; *r*0; / ↑ / 0 >	*CFdrd*
58	< 0; 1; *r*0; / ↑ / 0 >	
59	< 1; 0; *r*0; / ↑ / 0 >	
60	< 1; 1; *r*0; / ↑ / 0 >	
61	< 0; 0; *r*1; / ↓ / 1 >	
62	< 0; 1; *r*1; / ↓ / 1 >	
63	< 1; 0; *r*1; / ↓ / 1 >	
64	< 1; 1; *r*1; / ↓ / 1 >	
65	< 0; 0; *r*0; / 0 / 1 >	*CFir*
66	< 0; 1; *r*0; / 0 / 1 >	
67	< 1; 0; *r*0; / 0 / 1 >	
68	< 1; 1; *r*0; / 0 / 1 >	
69	< 0; 0; *r*1; / 1 / 0 >	
70	< 0; 0; *r*1; / 1 / 0 >	
71	< 1; 0; *r*1; / 1 / 0 >	
72	< 1; 1; *r*1; / 1 / 0 >	

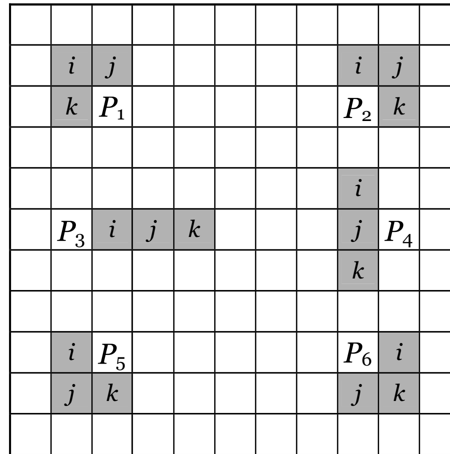

Fig. 1. Patterns for three physically adjacent cells.

This model of three-cell coupling, which comprises only physically adjacent memory cells, is called *reduced three-cell coupling.* Under this hypothesis and for the cases in which the mapping from logical addresses to physical cell locations is known completely, they have devised a march test of length $38n$, which covers this reduced model of three-cell coupling faults. A new more efficient march test with $30n$ operations (*MT-R3CF*) dedicated to the same reduced model of three-cell coupling faults is reported by Caşcaval, Bennett, and Huţanu [12]. But, all these test algorithms assume that a memory fault can be sensitized only by a transition write operation into a cell. Based on the model of all static two-cell coupling faults defined by Hamdioui, van de Goor and Rodgers in [7], this model of three-cell coupling faults has been extended by considering other classes of faults, such as the faults sensitized by a read or a non-transition write operation [1]. To cover this large fault model, called *all static reduced three-cell coupling faults*, the memory test $MarchS_3C$ has been proposed. This test is presented briefly in the following section.

4. The Memory Test $MarchS_3C$

The march tests are the most popular and widely accepted deterministic test algorithms because of their low temporal complexity, regular structures and their ability to detect a wide variety of memory faults. Usually, a complete march test is delimited by '$\{\dots\}$' bracket pair, while a march element is delimited by the '$(\dots)$' bracket pair. March elements are separated by semicolons, and the operations within a march element are separated by commas. Note that all operations of a march element are performed at a certain address, before proceeding to the next address. The whole memory is checked homogeneously in either one of two orders: ascending address order ($\Uparrow$) or descending address order ($\Downarrow$). When the address order is not relevant, the symbol $\Updownarrow$ is used.

The memory test we have considered for a possible BIST implementation is $MarchS_3C$. This memory test dedicated to the reduced model of all static three-cell coupling is presented in Fig. 2, where I_1, I_2, I_3, and I_4 are sequences which initialize the memory as follows: I_1 initializes the odd columns with 0 and the even columns with 1, and I_3 vice versa (*column-stripe data background*); I_2 and I_4 initialize the memory with a *checkerboard data background* and its complement, as illustrated in Fig. 3.

$$
\begin{aligned}
&\{ \Updownarrow (wo)^{(0)};\\
&\Uparrow (r, r, w, r, w_c)^{(1)}; \Uparrow (r, r, w, r, w_c)^{(2)}; \Downarrow (r, r, w, r, w_c)^{(3)}; \Downarrow (r, r, w, r, w_c)^{(4)};\\
&I_1^{(5)}; \Uparrow (r, r, w, r, w_c, r, r, w, r, w_c)^{(6)}; I_2^{(7)}; \Uparrow (r, r, w, r, w_c, r, r, w, r, w_c)^{(8)};\\
&I_3^{(9)}; \Uparrow (r, r, w, r, w_c, r, r, w, r, w_c)^{(10)}; I_4^{(11)}; \Uparrow (r, r, w, r, w_c, r, r, w, r, w_c)^{(12)};\\
&\Updownarrow (ro)^{(13)} \}
\end{aligned}
$$

Fig. 2. The memory test $MarchS_3C$.

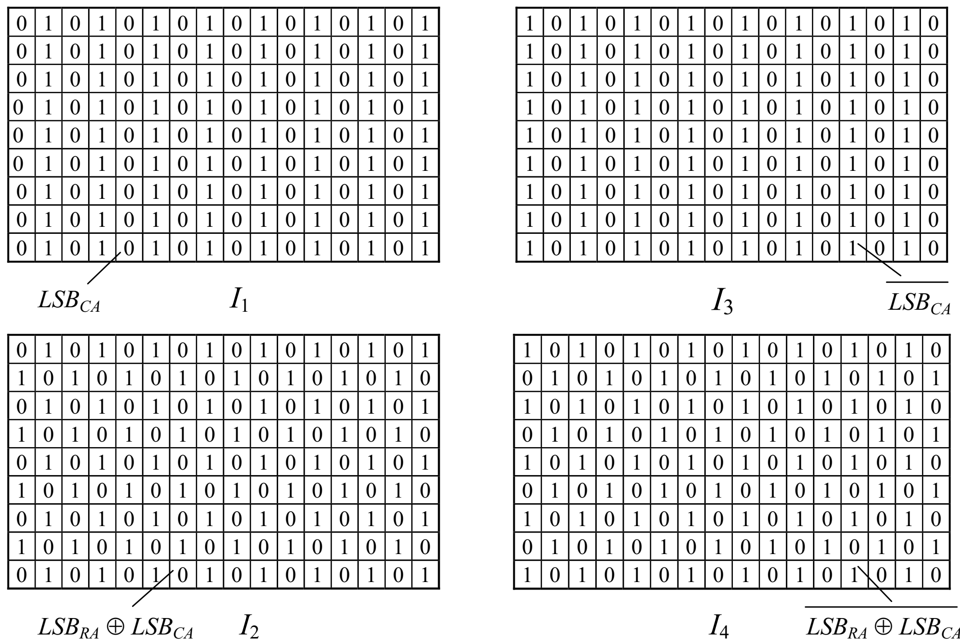

Fig. 3. Memory initialization for the memory test $MarchS_3C$.

This march test contains fourteen sequences as identified with a superscript (x), where $x \in \{0, 1, \ldots, 13\}$. The test sequences (5)-(12) form an alternating series of

background changes and march elements (as Cockburn proposed in [10]). Note that when changing from one background to the next, only the cells that must change states are written. Also, each write operation is preceded by a read operation. We can observe in Fig. 4 that any background change affects only a half of the cells. Each test sequence I_1, I_2, I_3 or I_4 performs $\frac{n}{2}$ read operations and $\frac{n}{2}$ write operations. Consequently, $MarchS_3C$ needs $66n$ operations. This march memory test is able to detect all the 72 FPs presented in Table 1 for all the six patterns illustrated in Fig. 1.

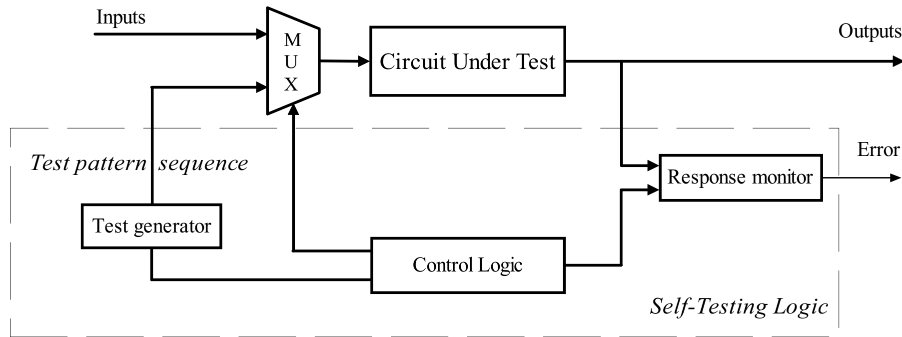

Fig. 4. Generic BIST scheme.

Regarding this march test, note that symbol $\Uparrow$ denotes an increasing address order, from address 0 to n-1, as long as symbol $\Downarrow$ denotes a reverse address order, from address n-1 to 0. Other permutations of the set of memory cell addresses decrease the effectiveness of the march test.

5. Simulation Results

To compare $MarchS_3C$ with other published tests, simulation results are presented in this section. The following march tests have been considered for the simulation study:

a) MT-R3CF [12] : $\{\Updownarrow(w0);\Uparrow(r,w1);\Uparrow(r,w0);\Downarrow(r,w1);\Downarrow(r,w0);\ I_1;$ $\Uparrow(r,w\mathrm{c},r,w\mathrm{c});\ I_2;\ \Uparrow(r,w\mathrm{c},r,w\mathrm{c});\ I_3;\ \Uparrow(r,w\mathrm{c},r,w\mathrm{c});\ I_4;\ \Uparrow(r,w\mathrm{c},r,w\mathrm{c});\ \Updownarrow(r)\}$, where I_1, I_2, I_3 and I_4 are sequences that initialize the memory as illustrated in Fig. 3.

b) Algorithm A [8] : $\{\Uparrow(w0);\ \Uparrow(r0,w1);\ \Uparrow(r1);\ \Uparrow(r1,w0);\ \Uparrow(r0);\ \Downarrow(r0,w1);\ \Downarrow(r1);$ $\Downarrow(r1,w0);\ \Downarrow(r0);\ \Uparrow(r0,w1,w0);\ \Uparrow(r0);\ \Downarrow(r0,w1,w0);\ \Downarrow(r0);\ \Uparrow(w1);\ \Uparrow(r1,w0,w1);$ $\Uparrow(r1);\ \Downarrow(r1,w0,w1);\ \Downarrow(r1)\ \}$.

c) March G [13] : $\{\Updownarrow(w0);\ \Uparrow(r0,w1,r1,w0,w1);\ \Uparrow(r1,w0,r0,w1);$ $\Downarrow(r1,w0,w1,\ w0);\ \Downarrow(r0,w1,r1,w0);\ \Uparrow(r0,w1,r1);\ \Downarrow(r1,w0,r0)\}$.

d) March S2C [14] : {$\Updownarrow(w0)$;$\Uparrow(r0,w1,r1,r1,w1)$; $\Uparrow(r1,w0,r0,r0,w0)$; $\Downarrow(r0,w1,r1,r1,w1)$; $\Downarrow(r1,w0,r0,r0,w0)$; $\Updownarrow(r0)$; }.

e) March LA [15] : {$\Updownarrow(w0)$; $\Uparrow(r0,w1,w0,w1,r1)$; $\Uparrow(r1,w0,w1,w0,r0)$; $\Downarrow(r0,w1,w0,w1,r1)$; $\Downarrow(r1,w0,w1,w0,r0)$; $\Downarrow(r0)$}.

f) March B [16] : {$\Updownarrow(w0)$; $\Uparrow(r0,w1,r1,w0,r0,w1)$; $\Uparrow(r1,w0,w1)$; $\Downarrow(r1,w0,w1,w0)$; $\Downarrow(r0,w1,w0)$}.

g) March LR [17] : {$\Updownarrow(w0)$; $\Downarrow(r0,w1)$; $\Uparrow(r1,w0,r0,w1)$; $\Uparrow(r1,w0)$; $\Uparrow(r0,w1,r1,w0)$; $\Uparrow(r0)$}.

h) March U [18] : {$\Updownarrow(w0)$;$\Uparrow(r0,w1,r1,w0)$;$\Uparrow(r0,w1)$; $\Downarrow(r1,w0,r0,w1)$; $\Downarrow(r1,w0)$}.

All the six patterns P_1, P_2, ..., P_6 illustrated in Fig. 1, and all FPs presented in Table 1 have been considered in this simulation study. Moreover, for each group of coupled cells, composed of two aggressor cells ($a1$-cell and $a2$-cell) and a victim cell (v-cell), six distinct combinations on the set $\{a1,\ a2,\ v\}$ have been considered. Consequently, 2952 (6×6×72) FPs have been simulated in order to evaluate the fault coverage of this reduced model of three-cell coupling faults. The simulation results are presented in Table 2.

Table 2. Fault coverage of this reduced model of three-cell coupling faults

#	Memory test	Test length	Fault coverage (%)
1	MarchS3C	66*n*	100
2	MT_R3CF	30*n*	56.73
3	Algorithm A	30*n*	46.30
4	March G	24*n*	44.91
5	MarchS2C	22*n*	53.40
6	March LA	22*n*	42.82
7	March B	17*n*	32.18
8	March LR	14*n*	43.52
9	March U	13*n*	42.59

As shown in Table 2, only $MarchS_3C$ is able to detect all FPs of simple three-cell coupling faults.

6. A BIST Logic Design for $MarchS_3C$

Generally, a circuit with BIST facilities has two operation modes: normal operation and test. In normal operation, the circuit receives its inputs from other modules

and performs the function for which it was designed. In the test mode, a test generator applies a sequence of test patterns to the memory, and a response monitor evaluates the test responses as illustrated in Fig. 4.

For the memory test $MarchS_3C$, a BIST logic with the block diagram presented in Fig. 5 is proposed.

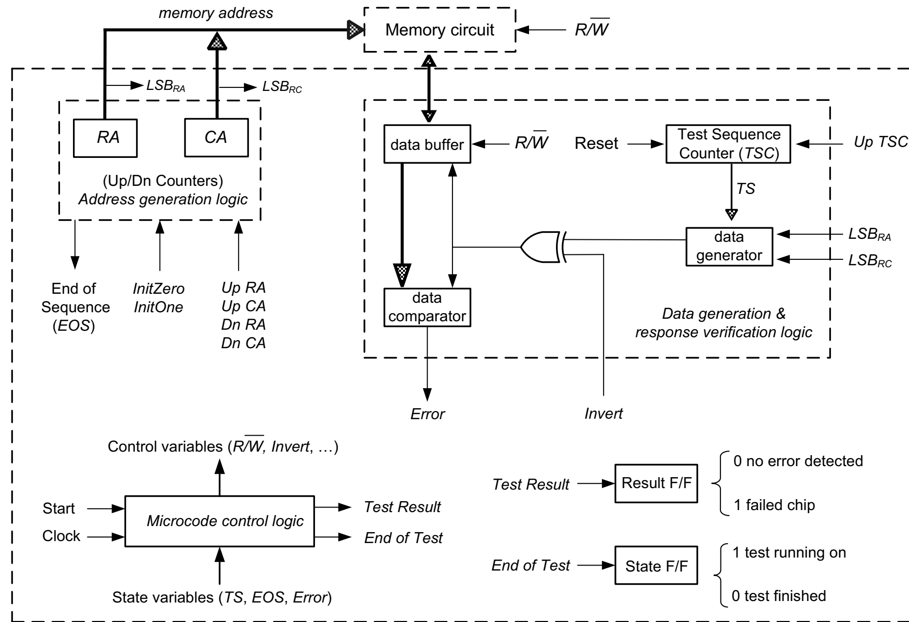

Fig. 5. The block diagram of BIST logic for the memory test $MarchS_3C$.

The BIST logic is composed of three parts: address–generation logic, data–generation and response–verification logic, and microcode control logic.

A. Address–generation logic

The address generation logic is composed of two up/down counters, for row address (RA) and column address (CA), respectively. Each address counter can be initialised with 0 (*InitZero*) or 1 (*InitOne*) in all bit locations. The memory is checked in ascending or descending order, so that, depending on the test sequence (TS), the control logic increments (Up) or decrements (Dn) one of the address counters, RA or CA.

B. Data generation and response-verification logic

The data-generation logic supplies data to be written in the memory and the expected data for response monitoring. An unique logic to generate both data for writing operations and expected data for response monitoring can be used. Note that, except on the first initialisation ($w0$) and the final checking of the memory ($r0$), the test algorithm is composed of two kinds of march elements, (r, r, w, r, w_c) and (r, r, w,

r, w_c, r, r, w, r, w_c). Every write operation into a cell is preceded by a read operation. The expected data in the first read operation of a cell is presented in Table 3, where *LSBCA* and *LSBRA* denote the less significant bit of the column address of the cell, and the less significant bit of the row address of the cell, respectively (Fig. 3).

Table. 3. Expected data in the first read operation of a cell

Test sequence (TS)	*Expected data*
(1)	0
(2)	1
(3)	0
(4)	1
(5)	0
(6)	LSB_{CA}
(7)	LSB_{CA}
(8)	$LSB_{CA} \oplus LSB_{RA}$
(9)	$LSB_{CA} \oplus LSB_{RA}$
(10)	$\overline{LSB_{CA}}$
(11)	$\overline{LSB_{CA}}$
(12)	$\overline{LSB_{CA} \oplus LSB_{RA}}$
(13)	$\overline{LSB_{CA} \oplus LSB_{RA}}$

Data generator is basically composed of a multiplexor with input variables in accordance with the expected data presented in Table 3, and a test sequence counter (*TSC*) that supplies the selection inputs. The logic variable *Invert* is used to command the XOR gate for changing the data bit generated.

C. Microcode control logic

The control logic initiates and controls the testing process. The control flow of the test algorithm $MarchS_3C$ is presented in Fig. 6.

In Fig. 6 the following notations are used to denote specific operations performed on the current memory cell:

- O ($w0$) – for the first initialization;
- O (*shortME*) – for the march element (r, r, w, r, w_c);
- O (*longME*) – for the march element (r, r, w, r, w_c, r, r, w, r, w_c);
- O ($r0$) – for the final checking.

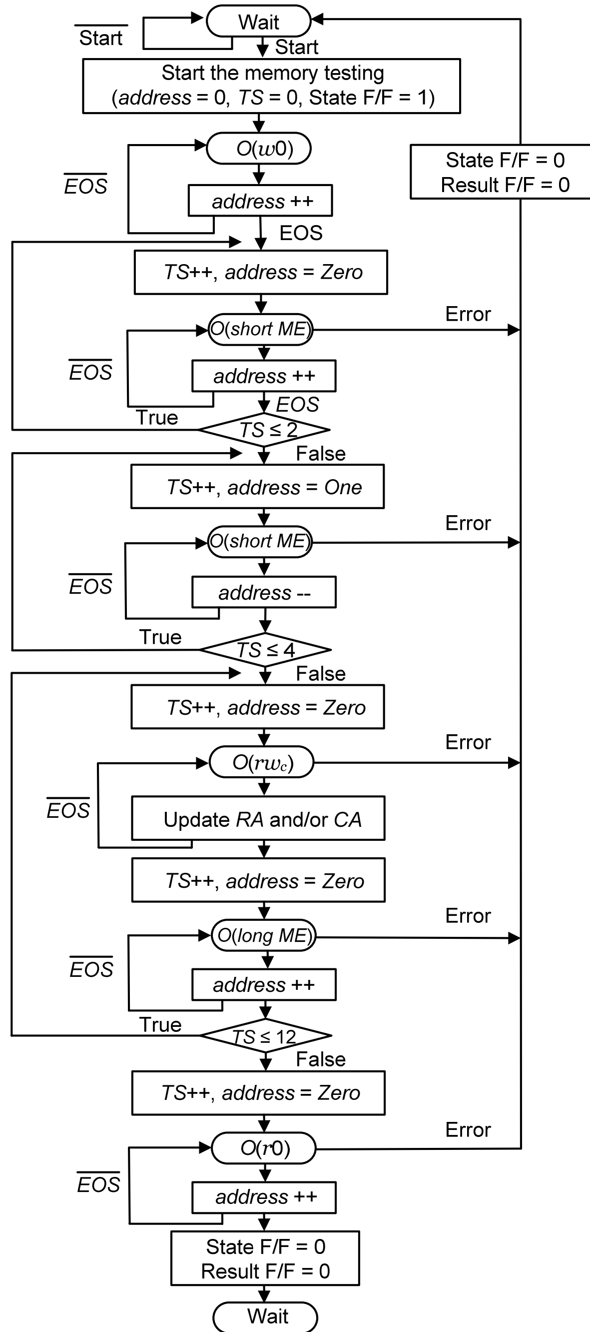

Fig. 6. Control flow for the memory test $MarchS_3C$.

7. Final Remarks

We are unaware of other memory test able to detect all FPs of this large model of three-cell coupling faults. To cover these harder-to-detect faults, $MarchS_3C$ uses multiple data backgrounds: a solid, a column−stripe, and a checkerboard. Although $MarchS_3C$ uses different patterns for memory initialization, the logic design we have proposed for a possible built-in self-testing implementation is not so complicated.

Regarding the BIST logic design for a MASB architecture, another response verification method is comparing the outputs of symmetrically placed bits in the tested arrays. An advantage of the parallel comparison method is that the expected values need not be generated.

References

[1] CAŞCAVAL P., CAŞCAVAL D., *March Test for a Reduced Model of All RAM Static 3-Cell Coupling Faults*, Bul. Inst. Polit. Iaşi, Tom **LIII (LVII)**, Autom. and Comput., pp. 87–96, 2007.

[2] ALLAN A. *et al.*, *2001 International Technology Roadmap for Semiconductors*, Computers, **35** (1), pp. 42–53, 2002.

[3] HAMDIOUI S., AL-ARS Z., JIMENEZ J., CALERO J., *PPM Reduction on Embedded Memories in System on Chip*, *Proc. IEEE European Test Symp.* (ETS'07), Freiburg, Germany, May 2007, pp. 85–90.

[4] HAMDIOUI S., *Testing Static Random Access Memories: Defects, Fault Models and Test Patterns*, *Kluwer Academic Publishers*, Norwell, USA, 2004.

[5] ADAMS R.D., *High Performance Memory Testing*, *Kluwer Academic Publishers*, Norwell, USA, 2003.

[6] FRANCKLIN M., SALUJA K., *Built-in Self Testing of RAMs*, IEEE Computer, **23** (10), pp. 45–56, 1990.

[7] HAMDIOUI S., VAN DE GOOR A.J., RODGERS M., *March SS: A Test for All Static Simple RAM Faults*, *Proc. of 10th IEEE Int. Workshop on Memory Technology, Design and Testing*, pp. 95–100, Isle of Bendor, France, July, 2002.

bibitemnair NAIR R., THATTE S., ABRAHAM J., *Efficient Algorithms for Testing Semiconductor Random-Access Memories*, IEEE Trans. on Computers, **C-27** (6), pp. 572–576, 1978.

[8] PAPACHRISTOU C., SAHGAL N., *An Improved Method for Detecting Functional Faults in Semiconductor Random-Access Memories*, IEEE Trans. on Computers, **C-34** (2), pp. 110–116, 1985.

[9] COCKBURN B.F., *Deterministic Testing for Detecting Single V-Coupling Faults in RAMs*, Journal of Electronic Testing, Theory and Applications, **5** (1), pp. 91–113, 1994.

[10] CAŞCAVAL P., BENNETT S., *Efficient March Test for 3-Coupling Faults in Random-Access Memories*, *Microprocessors and Microsystems*, **24** (10), pp. 501–509, 2001.

[11] CAŞCAVAL P., BENNETT S., HUŢANU C., *Efficient March Tests for a Reduced 3-Coupling and 4-Coupling Faults in Random-Access Memories*, Journal of Electronic Testing, Theory and Applications, **20** (3), pp. 227–243, 2004.

[12] VAN DE GOOR A.J., *Using March Tests to Test SRAMs*, IEEE Design and Test of Computers, **10** (1) pp. 8–14, 1993.

[13] CAŞCAVAL P., SILION R., STAN A., *MarchS2C: A Test for All Static 2-Cell RAM Coupling Faults*, Bul. Inst. Polit. Iaşi, Tom. **LII (LVI)**, Fasc. 1–4, Autom. and Comput., pp. 79–86, 2006.

[14] VAN DE GOOR A.J. *et al.*, *March LA: A Test for Linked Memory Faults*, *Proc. of European Design and Test Conference*, pp. 627–634, Paris, France, 1999.

[15] SUK D., REDDY S., *Test Procedures for a Class of Pattern-Sensitive Faults in Semiconductor Random-Access Memories*, IEEE Trans. on Computers, **C-29** (6), pp. 419–429, 1980.

[16] YARMOLIK V.N., VAN DE GOOR A.J., GAYDADJIEV G.N., MIKITJUK V.G., *March LR: A test for realistic linked faults*, *Proc. 14th IEEE VLSI Test Symp.* (VTS'96), pp. 272–280, 1996.

[17] VAN DE GOOR A.J., GAYDADJIEV G.N., *March U: A Test for All Unlinked Memory Faults*, *IEE Proc. of Circuits Devices and Systems*, **144** (3), pp. 155–160, 1997.