March AB, March AB1: new March tests for unlinked dynamic memory faults

(Article begins on next page)

28 April 2024

# March AB, March AB1:
# New March Tests for Unlinked Dynamic Memory Faults

A. Benso, A. Bosio, S. Di Carlo, G. Di Natale, P. Prinetto

Politecnico di Torino
Dipartimento di Automatica e Informatica
Torino, Italy
E-mail {benso, bosio, dicarlo, dinatale, prinetto}@polito.it
http://www.testgroup.polito.it

## Abstract

*Among the different types of algorithms proposed to test Static Random Access Memories (SRAMs), March Tests have proven to be faster, simpler and regularly structured. New memory production technologies introduce new classes of faults usually referred to as Dynamic Memory Faults. A few March Tests for dynamic fault, with different fault coverage, have been published. In this paper we propose new March Tests targeting Unlinked Dynamic Faults with lower complexity than published ones. Comparison results show that the proposed March Tests provide the same fault coverage of the known ones, but they reduce the test complexity, and therefore the test time*

## 1.     Introduction

Silicon area is now so cheap and integration technologies so advanced that one can embed in a *System-On-a-Chip* (SOC) all the components and functions that historically were placed on a hardware board. Within SOCs memories are the densest components.     The International Technology Roadmap for Semiconductors 2004 [1] forecasts that embedded memories will reach 90% of chips area surface in ten years. It is thus common finding, on a single chip, tens of memories of different types, sizes, access protocols and timing; moreover they can recursively be embedded in embedded cores.

Memory defects strongly depend on the target technology. Every time a new technology is introduced, new defects appear and new fault models must be defined. In the last years, the so called *Static Faults* (e.g., stuck-at faults, coupling fault ...) [2] have been the predominant fault type. They are characterized by being sensitized by the execution of just a single memory operation. New faulty behaviors occur in latest technologies [3] [4]. As an example, a write operation on a memory cell, immediately followed by a read operation, may cause the cell to flip. On the contrary, if just a single write or a single read, or a read which does not immediately follow a write are performed, the cell does not flip. These behaviors cannot be modeled as Static Faults since they require more than one operation to be sensitized, and are referred to as *Dynamic Faults*. The set of possible Dynamic Faults is theoretically unlimited and wherever a new fault is observed a new custom test algorithm has to be designed.

Although ad-hoc testing strategies are needed to address the peculiar set of faults that can affects SRAMs, their regular structure allows adopting particularly simple algorithms, the most popular one being *March Tests* [5]. While several March Tests targeting Static Faults have been proposed [6] [7] [8] [9] [10] [11] [2], few March Tests have been developed to detect Dynamic Faults. [12] presents March RAW1 and RAW, of length $13n$ and $26n$, respectively; the former one covers single-cell dynamic faults whereas the latter one detects two-cell dynamic fault (See Section 2). In [13] a modified March C- of length $10n$ is presented, to cover Dynamic Read Destructive Faults (See Section 2). This test must be customized resorting to the knowledge of the physical layout of the memory under test, in order to modify the address order of each March Element.

In the present paper we introduce two new March Tests (March AB, March AB1) targeting the same set of dynamic faults addressed by March RAW1 and RAW. March AB and March AB1 provide the same fault coverage of March RAW and RAW1 with a reduced complexity. To better identify the target faults, a taxonomy of possible dynamic faults is presented, and each addressed fault is modeled resorting to the Fault Primitive formalism introduced in [14].

To systematically prove the efficiency of the proposed March Tests, for each fault model the coverage conditions, i.e., the sequence of memory operations needed to sensitize and detect the fault effects, have been defined. We thus proved that both March AB and March AB1 respect the coverage conditions for each fault in their fault list. Furthermore, we compared their fault coverage with already published algorithms. The correctness of the proposed tests has been also proved by fault simulation experiments performed using an in-house developed memory fault simulator [15].

The paper is structured as follows: Section 2 introduces the Fault Primitive formalism and the dynamic fault taxonomy; Section 3 presents the new March Tests. Section 4 details the complete list of the coverage conditions. Comparisons evaluations are reported in Section 5, while Section 6 summarizes the main contributions and outlines future research activities.

## 2. Dynamic Faults Taxonomy

Usually, for test purposes, faults in memories are modeled as Functional Faults. A *Functional Fault Model* (FFM) is a deviation of the memory behavior from the expected one under a set of performed operations [14]. A FFM usually involves one or more *Faulty Memory Cells* (FC) classified in two categories: *Aggressor cells* (*a*-cells), i.e., the memory cells that sensitize a given FFM and *Victim cells* (*v*-cells), i.e., the memory cells that show the effect of a FFM. A *Dynamic Fault* (DF) is a FFM that requires two or more memory operations, *sequentially* applied, in order to be sensitized. FFM can be described by a set of Fault Primitives (FPs) [14]. A Fault Primitive is a triplet <S/F/R> representing the difference between an expected (good) and the observed (faulty) memory behavior where: S is a sequence of $m$ operations needed to sensitize the given fault, F is the faulty behavior, i.e. the value (state) stored in the victim cells after applying S, and R is the sequence of values read on the aggressor cell when applying S. Dynamic Faults are modeled by FPs with $m > 1$. The cardinality of the *Dynamic Fault Set* (DFS) is infinite, being the number of possible operations not limited. The DFS is usually split in subsets, each including the FFMs requiring the same number of operations to be sensitized. Thus "2-operations DFS" identifies the set of DFs that require the application of two memory operations to be sensitized ($m = 2$) whereas the "*m*-operation DFS" corresponds to a generic dynamic fault set.

It has been proved that the probability of a DF decreases when $m$ increase [16]. 2-operations DFs are the most popular in state-of the-art memories. Thus, in the sequel, we will focus on 2-operations DFS, only. We also focus on unlinked DFs, thus assuming each FFM be independent from each other. FFMs belonging to the 2-operations DFS can be additionally clustered according to the number of Faulty Memory Cells (#FC) involved in the fault. Two main categories are thus possible. The former one is characterized by #FC = 1 (single-cell 2-operations DFS); in this case the same memory cell acts as *a*-cell and *v*-cell at the same time. The latter one occurs when #FC≥2 (two-or-more-cells fault). We restrict the second category by considering #FC=2 (two-cells 2-operations DFS) since it is the most relevant one when the *a*-cell address differs from *v*-cell address [12].

The Single-cell 2-operations DFS is characterized by #FC=1 and $m$=2. By considering all the possible combination of operations, one obtains an amount of 30 FPs, representing the fault space. In [14] each FP has been verified using simulation, obtaining three different groups of realistic FPs corresponding to three different FFMs: *Dynamic Read Disturb Fault* (**dRDF**), where a write operation immediately followed by a read changes the logical value stored in the memory cell and returns an incorrect output; *Dynamic Deceptive Read Disturb Fault* (**dDRDF**) where a write operation immediately followed by a read changes the logical value stored in the memory cell, but returns the expected output; and *Dynamic Incorrect Read Disturb Fault* (**dIRF**) in which a write operation immediately followed by a read does not change the logical value stored in the memory cell, but returns an incorrect output. Table 1 shows the FPs that model these FFMs. Left and right columns report the name of the dynamic FFM and the list of FPs, respectively. Each FFM is composed of 4 FPs.

Two-cells 2-Operations DFS is characterized by #FC=2 and $m$=2. In this case, we have to distinguish between how many operations are applied on the *a*-cell and how many on the *v*-cell. Moreover, we have to consider the mutual position of *aggressor* and *victim* cell, e.g., if "*a*<*v*" or "*v*>*a*", where "*a* < *v*" means that the address of the *a*-cell is lower than the address of the *v*-cell. An exhaustive list of 192 FPs that covers all the 2-operations two-cells DFS is given in [14]. Only a subset of these has been demonstrated to be realistic [14] [12] and we shall focus on this subset, only. The functional fault models considered are:

- Dynamic Disturb Coupling Fault (dCFds) where a write operation followed immediately by a read operation performed on the a-cell causes the v-cell to flip;

- Dynamic Read Disturb Coupling Fault (dCFrd) where a write operation immediately followed by a read performed on the v-cell when the a-cell is in a given

state changes the logical value stored in the memory cell, and return an incorrect output;

- Dynamic Deceptive Read Disturb Coupling Fault (dCFdrd) in which a write operation immediately followed by a read performed on the v-cell when the a-cell is in a given state changes the logical value stored in the memory cell, but return the expected output.Dynamic Incorrect Read Disturb Coupling Fault (dCFir) where a write operation immediately followed by a read performed on the v-cell when the a-cell is in a given state does not affect the logical value stored in the memory cell, but returns an incorrect output.

Table 2 shows the FPs that compose each FFM; each functional fault is composed of 8 FP.

| FFM | Fault Primitives |
|---|---|
| dRDF | $<0w_0r_0/1/1>,<1w_1r_1/0/0>,<0w_1r_1/0/0>,<1w_0r_0/1/1>$ |
| dDRDF | $<0w_0r_0/1/0>,<1w_1r_1/0/1>,<0w_1r_1/0/1>,<1w_0r_0/1/0>$ |
| dIRF | $<0w_0r_0/0/1>,<1w_1r_1/1/0>,<0w1r1/1/0>,<1w_0r_0/0/1>$ |

**Table 1: Single-cell 2-operations Dynamic FFM**

| FFM | Fault Primitives |
|---|---|
| dCFds | $<0w_0r_0;0/1/->,<0w_0r_0;1/0/->,<1w_1r_1;1/0/->,<1w_1r_1;0/1/->$ <br> $<0w_1r_1;0/1/->,<1w_0r_0;1/0/->,<0w_1r_1;1/0/->,<1w_0r_0;0/1/->$ |
| dCFrd | $<0;0w_0r_0/1/1>,<1;0w_0r_0/1/1>,<1;1w_1r_1/0/0>,<0;1w_1r_1/0/0>$ <br> $<0;0w_1r_1/0/0>,<1;0w_1r_1/0/0>,<1;1w_0r_0/1/1>,<0;1w_0r_0/1/1>$ |
| dCFdrd | $<0;0w_0r_0/1/0>,<1;0w_0r_0/1/0>,<1;1w_1r_1/0/1>,<0;1w_1r_1/0/1>$ <br> $<0;0w_1r_1/0/1>,<1;0w_1r_1/0/1>,<1;1w_0r_0/1/0>,<0;1w_0r_0/1/0>$ |
| dCFir | $<0;0w_0r_0/0/1>,<1;0w_0r_0/0/1>,<1;1w_1r_1/1/0>,<0;1w_1r_1/1/0>$ <br> $<0;0w_1r_1/1/0>,<1;0w_1r_1/1/0>,<1;1w_0r_0/0/1>,<0;1w_0r_0/0/1>$ |

**Table 2: Two-cells 2-operations Dynamic FFM**

## 3. March Tests

This section presents the two new March Tests AB and AB1 targeting the FFMs introduced in Section 2.

As pointed out in [5], a *March Test* is a test algorithm composed of a sequence of *March Elements* (ME). Each *March Element* is a sequence of memory operations applied sequentially on a certain memory cell before proceeding to the next one. The way in which one moves from a certain address to another one is called *Address Order* (AO). The AO characterizes each ME.

Not necessarily an up/down AO means that the ME starts from the lowest/highest memory address to the highest/lowest address; one can choose an arbitrary AO as increasing AO, without reducing the fault coverage of a given March Test [17]; the constraints is that the decreasing AO must be the exactly reverse of the increasing AO. Hereinafter, we denote a March Test using a '{…}' bracket, and a March Element using a '(…)' bracket. The $i$-th operation is defined as $op_i$ where $op_i \in \{w_d, r_d\}$, $d \in \{0,1\}$ in which '$r_d$' means "read the

content of the memory cell and verify that its value is equal to $d$". The *complexity* of a March Test is defined as the number of memory operations included in it.

We obtained the new March Tests AB and AB1 by using the automatic March Test generation algorithm introduced in [18].

Figure 1 shows the **March AB1**, with a complexity of $11n$. March AB1 is able to detect faults belonging to the single-cell 2-operations DFS introduced in Section 2. Compared with the March RAW1, the state-of-the-art algorithm to target the same set of faults with a complexity of $13n$, March AB1 reaches the same results but reduces the test length by two operations.

Figure 2 shows the **March AB**, with a complexity of $22n$. March AB is able to detect the two-cells 2-operations DFS introduced in Section 2. If compared with March RAW ($26n$), the state-of-the-art algorithm to target two-cells 2-operations DFs, it provides the same fault coverage reducing the test length by 4 operations.

The generation process allows also the definition of a set of *Fault Coverage Conditions* (FCC) needed to cover each of the FFMs described in Section 2. Each FCC covers one or more FPs. It specifies the March Elements that a March Test has to include to detect the target FP. In the sequel of the paper we will introduce the coverage conditions for the set of dynamic faults listed in Section 2 to prove that the new generated March Tests satisfy all the conditions.

$$\{ \Updownarrow(w_0) \quad \Leftrightarrow (w_1,r_1,w_1,r_1,r_1) \quad \Updownarrow (w_0,r_0,w_0,r_0,r_0) \}$$
$$M1 \qquad\qquad M2 \qquad\qquad\qquad M3$$

**Figure 1: March AB1 O(n) = 11n**

$$\{ \Updownarrow(w_1) \Downarrow (r_1,w_0,r_0,w_0,r_0) \Downarrow (r_0,w_1,r_1,w_1,r_1) \Uparrow (r_1,w_0,r_0,w_0,r_0)$$
$$M1 \qquad M2 \qquad\qquad M3 \qquad\qquad M4$$
$$\Uparrow (r_0,w_1,r_1,w_1,r_1) \quad \Updownarrow (r_1) \}$$
$$M5 \qquad\qquad M6$$

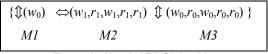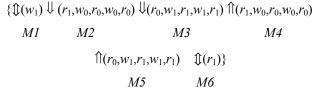**Figure 2: March AB O(n) = 22n**

## 4. Fault Coverage Condition

*Fault Coverage Conditions* (FCC) can be directly derived from the Functional Fault Model (FFM) definitions and in particular from the Fault Primitives (FPs) composing each FFM.

FCCs are expressed using the March Test notation by adding the following symbols:

- '…': any operation

- 'OP($d$)': every operations using the value $d$, $d \in \{0,1\}$ (e.g. OP(1) means $w_1$ or $r_1$)

- '(…)*': the operations included in bracket can be repeated 0 or more times

- '(…)+': the operations included in bracket can be repeated one or more times

- '[…]': includes one *relation* between two or more March Elements. A *relation* is specified by the logical operators between March Elements (AND, OR). As an example, the coverage condition [M1 M2 *OR* M3 M4] *AND* M5, where M$i$ is a March Element, means that the March Test has to include M1 M2 AND M5 or M3 M4 AND M5. Note that the sequence M$i$ M$j$ in a FCC means that the March Test must contain M$i$ immediately followed by M$j$.

## 4.1 Single-Cell Coverage Conditions

By analyzing the FPs introduced in Section 2, we can compact their notation as shown in Table 3, where $x \in \{0,1\}$ and $y=\text{not}(x)$.

| FFM | Fault Primitives | Fault Primitives (compact) |
|---|---|---|
| dRDF | $<0w_0r_0/1/1>,<1w_1r_1/0/0>,$ $<0w_1r_1/0/0>,<1w_0r_0/1/1>$ | $<xw_xr_x/y/y>$, $<xw_yr_y/x/x>$ |
| dDRDF | $<0w_0r_0/1/0>,<1w_1r_1/0/1>,$ $<0w_1r_1/0/1>,<1w_0r_0/1/0>$ | $<xw_xr_x/y/x>$, $<xw_yr_y/x/y>$ |
| dIRF | $<0w_0r_0/0/1>,<1w_1r_1/1/0>,$ $<0w_1r_1/1/0>,<1w_0r_0/0/1>$ | $<xw_xr_x/x/y>$, $<xw_yr_y/y/x>$ |

**Table 3: Single-cell 2-operations Dynamic FFM (compact notation)**

Figure 3 shows the list of FCCs for the single-cell 2-operations DFS covering FPs in Table 3.

As an example, FP1=$<0w_0r_0/1/1>$ is covered by FCC1. The March Test $\Updownarrow(w_0)$ $\Updownarrow(w_0,r_0)$ fulfill FCC1. The former March Element initializes the memory array to 0, whereas the latter one includes the operations $w_0,r_0$ that excite and observe FP1. In a similar way we can build a March Test for each FP.

Figure 3 shows that FP=$<xw_xr_x/x/y>$ is detected by FCC1 or FCC3 and FP=$<xw_yr_y/y/x>$ is detected by FCC2 or FFC4. Since each FFM is composed of several FPs, in order to cover the whole FFM the March Test must

include all the FCCs of the FPs composing the FFM. Table 4 reports, for each FFM (column 1), the related set of FPs (column 2) and, for each FP, the relative FFCs (column 3).

**FCC1**:

$[\Updownarrow((…)^*,OP(x))\ \Updownarrow(w_x,r_x,(…)^*)\ \text{or}\ \Updownarrow((...)^*,OP(x),w_x,r_x,(…)^*)]$

**FCC2**:

$[\Updownarrow((…)^*,OP(x))\ \Updownarrow(w_y,r_y,(…)^*)\ \text{or}\ \Updownarrow((...)^*,OP(x),w_y,r_y,(…)^*)]$

**FCC3**:

$[\Updownarrow((...)^*,OP(x))\ \Updownarrow(w_x,r_x,r_x,(…)^*)\ \text{or}\ \Updownarrow((...)^*,OP(x))\ \Updownarrow(w_x,r_x)\ \Updownarrow(r_x,(…)^*)\ \text{or}\ \Updownarrow((...)^*,OP(x),w_x,r_x,r_x,(…)^*)\ \text{or}\ \Updownarrow((...)^*,OP(x),\ w_x,r_x)\ \Updownarrow(r_x(…)^*)]$

**FCC4**:

$[\Updownarrow((...)^*,OP(x))\ \Updownarrow((w_y,r_y)\ +,r_y,(…)^*)\ \text{or}\ \Updownarrow((...)^*,OP(x))\ \Updownarrow((w_y,r_y)+)\ \Updownarrow(r_y,(…)^*)\ \text{or}\ \Updownarrow((...)^*,OP(x),\ (w_y,r_y)^+,r_y,(…)^*)\ \text{or}\ \Updownarrow((...)^*,OP(x),\ (w_y,r_y)+)\ \Updownarrow(r_y,(…)^*)]$

**Figure 3: Single-Cell Fault Coverage Conditions**

| FFM | FP | FFC |
|---|---|---|
| dRDF | $<0w_0r_0/1/1>$ | FFC1 |
| | $<1w_1r_1/0/0>$ | FFC1 |
| | $<0w_1r_1/0/0>$ | FFC2 |
| | $<1w_0r_0/1/1>$ | FFC2 |
| dDRDF | $<0w_0r_0/1/0>$ | FFC3 |
| | $<1w_1r_1/0/1>$ | FFC3 |
| | $<0w_1r_1/0/1>$ | FFC4 |
| | $<1w_0r_0/1/0>$ | FFC4 |
| dIRF | $<0w_0r_0/0/1>$ | FFC1 or FFC3 |
| | $<1w_1r_1/1/0>$ | FFC1 or FFC3 |
| | $<0w_1r_1/1/0>$ | FFC2 or FFC4 |
| | $<1w_0r_0/0/1>$ | FFC2 or FFC4 |

**Table 4: Condition for detecting single-cell dynamic fault**

### 4.2. Two-cell coverage conditions

The coverage conditions for two-cell DFs are more complex than those for single-cells. The main reason is that one has to take in account the relation between aggressor and victim cells. As shown in Section 2, the possible relations are: "$a<v$" and "$v>a$"; each condition has to cover both of them. To tackle the problem we had to specify the address order in the MEs. The complete list of two-cell FCCs is shown in Figure 4.

**FCC5**:
[$\updownarrow$((…)*,OP(x))$\Uparrow$($r_x$,(…)*,$w_x$,$r_x$,(…)*) and $\updownarrow$((…)*,OP(x))$\Downarrow$($r_x$,(…)*,$w_x$,$r_x$,(…)*) ] or
[$\updownarrow$((…)*,OP(y))$\Uparrow$((…)*,OP(x),$w_x$,$r_x$)$\Uparrow$($r_x$,(…)*) and $\updownarrow$((…)*,OP(y))$\Downarrow$((…)*,OP(x),($w_x$,$r_x$)$\Downarrow$($r_x$,(…)*)]

**FCC6**:
[$\updownarrow$((…)*,OP(y))$\Uparrow$($r_y$,(…)*,OP(x),$w_x$,$r_x$) and $\updownarrow$((…)*,OP(y))$\Downarrow$($r_y$,(…)*,OP(x),$w_x$,$r_x$)] o*r*
[$\Uparrow$((…)*,OP(x),$w_x$,$r_x$,(…)*,OP(y))$\Uparrow$($r_y$,(…)*)and $\Downarrow$((…)*,OP(x),$w_x$,$r_x$,(…)*,OP(y))$\Downarrow$($r_y$,(…)*)]

**FCC7**:
[$\updownarrow$((…)*,OP(x)) $\Uparrow$($r_x$,(…,OP(x))*,$w_y$,$r_y$,(…)*) and $\updownarrow$((…)*,OP(x)) $\Downarrow$($r_x$,(… ,OP(x))*,$w_y$,$r_y$, (…)*)] or
[$\updownarrow$((…)*,OP(x)$\Uparrow$((…,OP(x))*,$w_y$,$r_y$,OP(x))$\Uparrow$($r_x$, (…)*) and $\updownarrow$((…)*,OP(x)$\Downarrow$((…,OP(x))*,$w_y$,$r_y$,OP(x))$\Downarrow$($r_x$, (…)*)]

**FCC8**:
[$\updownarrow$((…)*,OP(x) $\Uparrow$($r_x$,(…,OP(x))*,$w_y$,$r_y$,(…,OP(y))*) $\Uparrow$($r_y$,(…)*) and $\updownarrow$((…)*,OP(x) $\Downarrow$($r$x,(…,OP(x))*,$w_y$,$r_y$,(…,OP(y))*) $\Downarrow$($r_y$,(…)*)] or
[$\updownarrow$((…)*,OP(y)) $\Uparrow$((…)*,OP(x),$w_y$,$r_y$)$\Uparrow$($r_y$,(…)*,) and $\updownarrow$((…)*,OP(y)) $\Downarrow$((…)*,OP(x),$w_y$,$r_y$) $\Downarrow$($r_y$,(…)*)]

**FCC9**:
[$\updownarrow$((…)*,OP(x)) $\updownarrow$((…,OP(x))*,$w_x$,$r_x$,(…,OP(x))*)] or [$\updownarrow$((…)*,OP(x)) $\Uparrow$((…,OP(x))*,$w_x$,$r_x$,(…)*,OP(y)) and $\updownarrow$((…)*,OP(x)) $\Downarrow$((…,OP(x))*, $w_x$,$r_x$,(…)*,OP(y))] or [$\Uparrow$((…)*,OP(x), $w_x$,$r_x$,(…,OP(x))*) and $\Downarrow$((…)*,OP(x), $w_x$,$r_x$,(…,OP(x))*)]

**FCC10**:
[$\updownarrow$((…)*,OP(y)) $\updownarrow$((…)*,OP(x),$w_x$,$r_x$, (…)*,OP(y))] or [$\Uparrow$((…)*,OP(x),$w_x$,$r_x$, (…)*,OP(y)) and $\Downarrow$ ((…)*,OP(x),$w_x$,$r_x$, (…)*,OP(y))] or
[$\updownarrow$((…)*,OP(y)) $\Uparrow$((…)*,OP(x),$w_x$,$r_x$, (…)*) and $\updownarrow$((…)*,OP(y)) $\Downarrow$((…)*,OP(x),$w_x$,$r_x$, (…)*)]

**FCC11**:
[$\updownarrow$((…)*,OP(x)) $\Uparrow$((…,OP(x))*,$w_y$,$r_y$, (…)*) and $\updownarrow$((…)*,OP(x)) $\Downarrow$((…,OP(x))*,$w_y$,$r_y$, (…)*)] or [$\updownarrow$((…)*,OP(y)) $\Uparrow$((…)*,OP(x),$w_y$,$r_y$, (…)*,OP(x)) and $\updownarrow$((…)*,OP(y)) $\Downarrow$((…)*,OP(x),$w_y$,$r_y$, (…)*,OP(x))]

**FCC12**:
[$\updownarrow$((…)*,OP(y)) $\updownarrow$( (…)*,OP(x),$w_y$,$r_y$,(…,OP(y))*)] or [$\updownarrow$((…)*,OP(y)) $\Uparrow$((…)*,OP(x),$w_y$,$r_y$, (…)*) and $\updownarrow$((…)*,OP(y)) $\Downarrow$((…)*,OP(x),$w_y$,$r_y$, (…)*)] or [$\updownarrow$((…)*,OP(y)) $\Uparrow$((…)*,OP(x),$w_y$,$r_y$, (…)*) and $\updownarrow$((…)*,OP(y)) $\Downarrow$((…)*,OP(x),$w_y$,$r_y$, (…)*)]

**FCC13**:
[$\updownarrow$((…)*,OP(x)) $\updownarrow$((…,OP(x))*,$w_x$,$r_x$) $\updownarrow$($r_x$,(…)*)]or [$\updownarrow$((…)*,OP(x)) $\updownarrow$((…,OP(x))*,$w_x$,$r_x$,$r_x$, (…)*)]or [$\updownarrow$((…)*,OP(y)) $\Uparrow$((…)*,OP(x),$w_x$,$r_x$)$\Uparrow$($r_x$,(…)*) and $\updownarrow$((…)*,OP(y)) $\Downarrow$((…)*,OP(x),$w_x$,$r_x$) $\Downarrow$($r_x$, (…)*)]

**FCC14**:
[$\updownarrow$((…)*,OP(y)) $\updownarrow$((…)*,OP(x),$w_x$,$r_x$,$r_x$, (…)*,OP(y))]or [$\updownarrow$((…)*,OP(y)) $\Uparrow$((…)*,OP(x),$w_x$,$r_x$)$\Uparrow$ ($r_x$, (…)*) and $\updownarrow$((…)*,OP(y)) $\Downarrow$((…)*,OP(x),$w_x$,$r_x$) $\Downarrow$($r_x$,(…)*)]

**FCC15**:
[$\updownarrow$((…)*,OP(x)) $\Uparrow$((…,OP(x))*,($w_y$,$r_y$)$^+$) $\Uparrow$($r_y$,(…)*) and $\updownarrow$((…)*,OP(x)) $\Downarrow$((…,OP(x))*,($w_y$,$r_y$)$^+$) $\Downarrow$($r_y$,(…)*)] or [$\updownarrow$((…)*,OP(y)) $\Uparrow$((…)*,OP(x),$w_y$,$r_y$,$r_y$, (…)*,OP(x)) and $\updownarrow$((…)*,OP(y)) $\Downarrow$((…)*,OP(x),$w_y$,$r_y$,$r_y$, (…)*,OP(x))]

**FCC16**:
[$\updownarrow$((…)*,OP(y)) $\updownarrow$((…)*,OP(x),$w_y$,$r_y$,$r_y$,(…,OP(y))*)] or [((…)*,OP(x)) $\Uparrow$((…,OP(x))*,($w_y$,$r_y$)$^+$) $\Uparrow$($r_y$, (…)*) and $\updownarrow$((…)*,OP(x)) $\Downarrow$((…,OP(x))*,($w_y$,$r_y$)$^+$) $\Downarrow$($r_y$, (…)*)]

**Figure 4: Two Cell Fault Coverage Condition**

As for single cells, each FCC covers a single FP by specifying the March Element that a March Test must include to detect the target FP.

The dCFds (see Section 2) is composed of $<xw_x r_x;x/y/->$, $<xw_x r_x;y/x/->$, $<xw_y r_y;x/y/->$ and $<xw_y r_y;y/x/->$; dCFrd (Section 2) is composed of $<x;xw_x r_x/y/y>$, $<y;xw_x r_x/y/y>$, $<x;xw_y r_y/x/x>$ and $<y;xw_y r_y/x/x>$; dCFdrd (Section 2.2) is composed of $<x;xw_x r_x/y/x>$, $<y;xw_x r_x/y/x>$, $<x;xw_y r_y/x/y>$ and $<y;xw_y r_y/x/y>$; dCFir (Section 2.2) is composed of $<x;xw_x r_x/x/y>$, $<y;xw_x r_x/x/y>$, $<x;xw_y r_y/y/x>$ and $<y;xw_y r_y/y/x>$. FP=$<x;xw_x r_x/x/y>$ is detected by FCC9 or FCC13, FP=$<y;xw_x r_x/x/y>$ is detected by FCC10 or FFC14, FP=$<x;xw_y r_y/y/x>$ is detected by FCC11 or FFC15 finally FP=$<y;xw_y r_y/y/x>$ is detected by FCC12 or FFC16. Table 4 reports for each fault model (column 2) its set of FP (column 3) and for each FP the relative fault coverage conditions (column 4).

| FFM | FP | FFC |
|---|---|---|
| dCFds | $<xw_x r_x;x/y/->$ | FFC5 |
| | $<xw_x r_x;y/x/->$ | FFC6 |
| | $<xw_y r_y;x/y/->$ | FFC7 |
| | $<xw_y r_y;y/x/->$ | FFC8 |
| dCFrd | $<x;xw_x r_x/y/y>$ | FFC9 |
| | $<y;xw_x r_x/y/y>$ | FFC10 |
| | $<x;xw_y r_y/x/x>$ | FFC11 |
| | $<y;xw_y r_y/x/x>$ | FFC12 |
| dCFdrd | $<x;xw_x r_x/y/x>$ | FFC13 |
| | $<y;xw_x r_x/y/x>$ | FFC14 |
| | $<x;xw_y r_y/x/y>$ | FFC15 |
| | $<y;xw_y r_y/x/y>$ | FFC16 |
| dCFir | $<x;xw_x r_x/x/y>$ | FFC9 or FFC13 |
| | $<y;xw_x r_x/x/y>$ | FFC10 or FFC14 |
| | $<x;xw_y r_y/y/x>$ | FFC11 or FFC15 |
| | $<y;xw_y r_y/y/x>$ | FFC12 or FFC16 |

**Table 5: Condition for detecting two-cell dynamic faults**

## 5. Comparing March Tests

In this section we compare the proposed March Tests (March AB, March AB1) with the following published March Tests:

- March C- [19]: $\{ \Updownarrow(w_0) \Uparrow(r_0,w_1) \Uparrow(r_1,w_0) \Downarrow(r_0,w_1) \Downarrow(r_1,w_0) \Updownarrow(r_0)\}$
- March RAW1 [12]: $\{ \Updownarrow(w_0) \Updownarrow(w_0,r_0) \Updownarrow(r_0) \Updownarrow(w_1,r_1) \Updownarrow(r_1) \Updownarrow(w_1,r_1) \Updownarrow(r_1) \Updownarrow(w_0,r_0) \Updownarrow(r_0)\}$
- March SS [8]: $\{ \Updownarrow(w_0) \Uparrow(r_0,r_0,w_0,r_0,w_1) \Uparrow(r_1,r_1,w_1,r_1,w_0) \Downarrow(r_0,r_0,w_0,r_0,w_1) \Downarrow(r1,r1,w1,r1,w0) \Updownarrow(r0)\}$
- March RAW [12]: $\{ \Updownarrow(w_0) \Uparrow(r_0,w_0,r_0,r_0,w_1,r_1) \Uparrow(r_1,w_1,r_1,r_1,w_0,r_0) \Downarrow(r_0,w_0,r_0,r_0,w_1,r_1) \Downarrow(r_1,w_1,r_1,r_1,w_0,r_0) \Updownarrow(r_0)\}$

Each March algorithm has been simulated by using the memory fault simulator presented in [15]. Table 6 summarizes the fault coverage percentage obtained by every test.

In Table 7 and 8, for each FCC introduced in Section 4 and for each of the considered March Tests we identify the ME where the FCC appears.

Both tables use the following notation: 'M$i$' ($i=1,2…,z$) denote the $i$-th March Element, 'M$i$:M$j$' denotes that FCC occurs between $i$-th ME and $j$-th ME; instead 'M$i$,M$j$' denotes that the FCC occurrence appear both in M$i$ and M$j$.

We compare our tests with March RAW and March RAW1 [12] since they are explicitly designed to cover the same set of dynamic fault targeted by March AB and March AB1. We also consider March C- since in [13] using an address order customized on the physical layout of the memory it is used to detect dynamic faults. Nevertheless, in our experiments, we consider only the classic March C- to show that, without any information about the memory layout, the fault coverage becomes very marginal (around 0,39%) (Table 6).

Of particular interest is the comparison with March SS [8], originally designed to cover the full set of memory static faults. As shown in Table 6, March AB has the same complexity of March SS, it covers the same set of static faults but it covers dynamic faults, as well. Furthermore, due to its regular and symmetric structure, March AB becomes a natural candidate for memory BIST architectures, making our solution very attractive for the industry.

Comparison results show that the proposed March Tests provide the same fault coverage of the known ones, but they reduce the test complexity, and therefore the test time.

| MT | O(n) | Single-cell | | | Two-cell | | | |
|---|---|---|---|---|---|---|---|---|
| | | dRDF | dDRDF | dIRF | dCFds | dCFrd | dCFrdr | dCFir |
| C- | 10n | 0.39% | 0.39% | 0.39% | 0.83% | 0.39% | 0.39% | 0.39% |
| AB1 | 11n | 100% | 100% | 100% | 0% | 50% | 50% | 50% |
| RAW1 | 13n | 100% | 100% | 100% | 0% | 50% | 50% | 50% |
| SS | 22n | 25% | 0.39% | 25% | 50% | 50% | 0.39% | 50% |
| AB | 22n | 100% | 100% | 100% | 100% | 100% | 100% | 100% |
| RAW | 26n | 100% | 100% | 100% | 100% | 100% | 100% | 100% |

**Table 6: Simulation results**

| MT | O(n) | Single-cell | | | |
|---|---|---|---|---|---|
| | | FFC1 | FFC2 | FFC3 | FFC4 |
| C- | 10n | - | - | - | - |
| AB1 | 11n | M2,M3 | M1:M2, M2:M3 | M2,M3 | M1:M2, M2:M3 |
| RAW1 | 13n | M1:M2, 6:M7 | M3:M4,M8:M9 | M1:M2:M3,M5:M6:M7 | M3:M4:M5, M7:M8:M9 |
| SS | 22n | M2,M3,M4,M5 | - | - | - |
| AB | 22n | M2,M3,M4,M5 | M2,M3,M4,M5 | M2:M3,M3:M4,M4:M5 ,M5:M6 | M2:M3,M3:M4,M4:M5 ,M5:M6 |
| RAW | 26n | M2,M3,M4,M5 | M2,M3,M4,M5 | M2,M3,M4,M5 | M2:M3,M4:M5,M5:M6 |

**Table 7: Evidence of single-cell FCCs**

| MT | O(n) | Two-cell (FFC) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| C- | 10n | - | - | - | - | - | - | - | - | - | - | - | - |
| AB1 | 11n | - | - | - | - | M2,M3 | - | - | - | M2,M3 | - | - | - |
| RAW1 | 13n | - | - | - | - | M1,M2 | - | - | - | M1:M2:M3 , M5:M6:M7 | - | - | - |
| SS | 22n | M1:M2, M2:M3, M3:M4, M4:M5 | M2,M3, M4,M5 | - | - | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | - | - | - | - | - | - |
| AB | 22n | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M2,M3 M4,M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 |
| RAW | 26n | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 | M1:M2, M2:M3, M3:M4, M4:M5 |

**Table 8: Evidence of two-cell FCCs**

## 6. Conclusions

This paper proposed two new March Tests targeting dynamic memory faults. March AB and March AB1 have the same fault coverage of already published algorithms (March RAW and March RAW1) addressing the same fault list but they reduce the complexity of 4 and 2 operations respectively. The proposed March Tests have been obtained by applying an automatic March Tests generation algorithm.

To prove the correctness of the new March Tests the coverage conditions needed to detect each fault in the fault list have been proposed.

In addition, the comparison between March AB and the well-known March SS for static fault shows that March AB has the same length of March SS but it improves the fault coverage by detecting both static and dynamic faults, making our solution very attractive industrially.

## 7.     References

[1]    International Technology Roadmap for Semiconductors, "International technology roadmap for semiconductors 2004 Update", http://public.itrs.net/Home.htm, 2004

[2]    A. J. van de Goor, "Testing Semiconductor Memories: theory and practice", *Wiley, Chichester (UK)*, 1991

[3]    S. Hamdioui, R. Wadsworth, J.D. Reyes, A. J. van de Goor, "Importance of Dynamic Faults for new SRAM Technologies", *ETW 2003, 8th IEEE European Test Workshop*, 2003, pp. 29 -34

[4]    Z. Al-Ars, Ad J. van de Goor, "Static and Dynamic Behavior of Memory Cell Array Opens and Shorts in Embedded DRAMs", *DATE 2001, IEEE Design Automation and Test in Europe*, 2001, pp. 496-503.

[5]    A. J. van de Goor, "Using March Tests to Test SRAMs", *IEEE Design & Test of Computers, Volume*: 10 Issue: 1, March 1993 pp: 8 –14.

[6]    R.D. Adams and E.S. Cooley, "Analysis of a Deceptive Destructive Read Memory fault Model and Recommended Testing", *NATW 1996. 5th IEEE North Atlantic Test Workshop*, 1996

[7]    R. Dekker, F. Beenker, L. Thijssen, "A Realistic Fault Model and Test Algorithms for Satic Random Acces Memory", *IEEE Transaction on Computer-Aided Design*, Volume: 9, Issue: 6, June 1990

[8]    S. Hamdioui, Ad J. van de Goor, M. Rodgers, "March SS: A Test for All Static Simple RAM Faults", MTDT 2002: *IEEE International Workshop on Memory Technology, Design and Testing*, 2002 pp. 95 – 100.

[9]    M. Marinescu, "Simple and Efficient Algorithms for Functional RAM Testing", *ITC 1982, IEEE International Test Conference,* 1982, pp. 236-239.

[10]   R. Nair, "An Optimal Algorithm for Testing Stuck-at Faults Random Access Memories", *IEEE Transaction on Computer*, Vol. C-28, No. 3, 1979, pp. 258-261.

[11]   D.S. Suk, S.M. Reddy, "A March Test for Functional Faults in Semiconductors Random-Access Memories", *IEEE Transaction on Computer*, Vol C-30, No. 12, 1981, pp. 982-985.

[12]   S. Hamdioui, Z. Al-Ars, A. J. van de Goor, "Testing Static and Dynamic Faults in Random Access Memories", *VTS 2002, 20th IEEE VLSI Test Symposium*, 2002, pp.

[13]   L. Dilillo,P. Girard, S. Pravossoudovitch, A. Virazel, S. Borri, M. Hage-Hassan, "Dynamic read destructive fault in embedded-SRAMs: analysis and march test solution", *ETS 2004, 9th IEEE European Test Symposium*, 2004, pp. 140-145.

[14]   A. J. van de Goor, Z. Al-Ars, "Functional Memory Faults: A Formal Notation and a Taxonomy", *VTS 2000, 18th IEEE VLSI Test Symposium*, 2000, pp. 281-289.

[15]   A. Benso, S. Di Carlo, G. Di Natale, P. Prinetto, "Specification and design of a new memory fault simulator", *ATS 2002, 11th IEEE Asian Test Symposium,* 2002.pp. 92 – 97.

[16]   Z. Al-Ars and A.J. van de Goor, "Approximating Infinite Dynamic Behavior for DRAM Cell Defects", *VTS 2002, 20th IEEE VLSI Test Symposium,* 2002, pp.401-406.

[17]   D. Niggemeyer, M. Redeker, J. Otterstedt, "Integration of non-classical faults in standard March tests", *MTDT 1998, IEEE International Workshop on Memory Technology, Design and Testing*, 1998, pp. 91 -96

[18]   A. Benso, A. Bosio, S. Di Carlo, G. Di Natale, P. Prinetto, "Automatic March Tests Generation for Static and Dynamic Faults in SRAMs", *ETS 2005, 10th IEEE European Test Symposium*, 2005.

[19]   M. Marinescu, "Simple and Efficient Algorithms for Functional RAM Testing", *Proc. Int. Test Conf.*, 1982, pp.236-239.