March AB, a State-of-the-Art March Test for Realistic Static Linked Faults and Dynamic Faults in SRAMs

March AB, a State-of-the-Art March Test for Realistic Static Linked Faults and Dynamic Faults in SRAMs

Authors: Bosio A., Di Carlo S., Di Natale G., Prinetto P.,

Published in IET COMPUTERS & DIGITAL TECHNIQUES Vol. 1, No. 3, 2007, pp. 237-245.

DOI: 10.1049/iet-cdt:20060137

March AB, a State-of-the-Art March Test for Realistic Static Linked Faults and Dynamic Faults in SRAMs

A. Bosio, S. Di Carlo, G. Di Natale, P. Prinetto
Politecnico di Torino
Dipartimento di Automatica e Informatica
Corso Duca degli Abruzzi 24, I-10129
Torino, Italy
E-mail:{ alberto.bosio, stefano.dicarlo, giorgio.dinatale, paolo.prinetto}@polito.it
http://www.testgroup.polito.it

Abstract

Memory testing commonly faces two issues: the characterization of detailed and realistic fault models, and the definition of time-efficient test algorithms able to detect them. Among the different types of algorithms proposed for testing Static Random Access Memories (SRAMs), march tests have proven to be faster, simpler and regularly structured. The continuous evolution of the memory technology requires the constant introduction of new classes of faults, such as dynamic and linked faults.

In this paper we present March AL – a march test targeting realistic memory static linked faults and dynamic unlinked faults. Comparison results show that the proposed march test provides the same fault coverage of already published algorithms reducing the test complexity and therefore the test time.

1 Introduction

Memories are one of the most important components in digital systems, and semiconductor memories are nowadays one of the fastest growing technologies. [1] forecasts that embedded memories will reach 90% of chips area surface in ten years. As a result, the production yield will depend largely on memories and the development of efficient test solutions and repair schemes for memories will be essential.

In the last years, so called unlinked static faults (e.g., stuck-at faults, coupling faults, etc.) [2] have been the predominant fault type. As technologies move to very deep submicron devices, new classes of faults appear to be more and more problematic from a test point of view. Among them, dynamic and linked faults are the most relevant [3].

Dynamic faults [3][4][5] require more than one operation to be sensitized. The set of possible dynamic faults is theoretically unlimited [6].

A linked fault is a memory fault composed of two or more simple faults. Each simple fault can be influenced by the remaining ones, and in some cases the fault can be masked.

Designing efficient tests to deal with both dynamic and linked faults is a challenge to assure the quality of future memory cores.

March tests remain the most attractive solution due to their linear complexity and effectiveness for detection of a large number of other faults [2]. While several march tests targeting static unlinked faults have been proposed [2][7][8][9][10], few of them have been developed to detect dynamic unlinked faults and static linked faults.

In [11] the authors present march RAW1 and RAW (of complexity 13n and 26n, respectively). The former one targets single-cell dynamic faults whereas the latter one detects two-cells dynamic faults.

In [12] a modified March C- of complexity 10n is presented, to cover a particular type of dynamic fault model called Dynamic Read Destructive Fault. The authors resort to the knowledge of the physical layout of the memory under test, in order to modify the address order of each march element and to obtain additional coverage.

March A, March B [10], March LA [13], and March LR [14] have an high fault coverage on a restricted set of linked memory faults. In [15] the authors present an automatically generated march test for static linked faults having complexity of 43n. This march test is still affected by the problem of detecting a limited number of static linked faults, as in [13][14].

In [16] and [17] the authors present an accurate analysis of the linked faults concept. They also present a march test facing new fault models. The presented March SL has a complexity of 41n.

In [18] the authors present March MSL that reduces the complexity of March SL to 23n without any loss in the fault coverage.

Despite the aoove march tests have high fault coverage when dealing with a single class of faults, they loose effectiveness when dealing with multiple class of faults.

In this paper we propose March AB, a march test able to detect both realistic dynamic unlinked faults and realistic static linked faults. Moreover, March AB still covers the whole set of realistic static unlinked faults proposed in [5]. To better identify what we consider as realistic dynamic and linked faults, a taxonomy of these new category of faults is presented.

To analytically prove the efficiency of the proposed march test, we will define for each fault model the coverage conditions, i.e. the sequence of memory operations needed to sensitize and detect the

fault effects. Then we will prove that March AB respects the coverage conditions for each fault in the fault list. Finally, we will compare the fault coverage with already published march tests.

The paper is structured as follows: Section 2 details the fault taxonomy, Section 3 introduces March AB and, Section 4 validates the proposed march test by introducing the concept of coverage conditions. Comparisons with already published solutions are reported in Section 5. Finally Section 6 summarizes the main contributions and outlines future research activities.

2 Fault model notation and taxonomy

Memory defects can manifest themselves in too many different ways, making impractical the creation of an exhaustive list of them. Thus in a typical test methodology an abstraction of the defects called functional fault model is used to generate and evaluate tests.

A Functional Fault Model (FFM) is a deviation of the memory behavior from the expected one under a set of performed operations. A FFM involves one or more Faulty Memory Cells (FMC) classified in two categories: (i) aggressor cells (*a*-cells), i.e., the memory cells that sensitize the FFM and (ii) victim cells (*v*-cells), i.e., the memory cells that show the effect of the FFM.

Functional fault models can be described using the Fault Primitive (FP) formalism introduced in [5]. A FP is identified by <S/F/R>. It represents the difference between an expected (good), and the observed (faulty) memory behavior of a memory device, where:

- *S* is a sequence of *m* operations, applied on the *a*-cells and *v*-cells, needed to sensitize the given fault;
- F is the faulty behavior, i.e., the value (state) stored in the *v*-cells after applying S;
- R describes the logic output level of a read operation (e.g., 0) in case S contains a read operation applied to the *a*-cell.

Several FPs classification rules can be adopted, based on the number of memory operations (*m*) needed to sensitize the FP, and based on the number of memory cells (#FMC) involved by the FP [5]. In the sequel of this section we present a collection of both dynamic and linked fault models that have been proved as the most realistic in modern technologies [5][11][17]. This list will represent the target fault list for our march test.

2.1 Dynamic fault taxonomy

Dynamic faults are modeled by FPs with $m>1$. The theoretical number of possible dynamic faults is infinite, being the number of possible operations not limited. The set of dynamic faults is usually split

in subsets, each one including FFMs requiring the same number of operations to be sensitized. As an example, two-operations dynamic faults require the application of two memory operations to be sensitized (m=2).

It has been proved that the probability of a dynamic fault decreases when m increases [6] and two-operations dynamic faults are the most popular in state-of-the-art memories. Thus, in this paper, we will focus on two-operations dynamic faults, only. We also focus on unlinked dynamic faults, thus assuming each FFM being independent from each other. FFMs modeling two-operations dynamic faults can be additionally clustered according to the number of faulty memory cells (#FMC) involved in the fault. We consider two main categories: (i) single-cell two-operations dynamic faults (#FMC=1), and (ii) two-cells two-operations dynamic faults (#FMC=2).

2.1.1 Single-cell two operations dynamic faults

Single-cell two-operations dynamic faults are characterized by #FMC=1 and m=2. The fault space is composed of all the possible combinations of two operations on the faulty cell, i.e. 30 different FPs. In [5] each FP has been verified by 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 operation changes the logical value stored in the faulty memory cell and returns an incorrect output;
- *Dynamic Deceptive Read Disturb Fault* (**dDRDF**), where a write operation immediately followed by a read operation changes the logical value stored in the faulty memory cell, but returns the expected output;
- *Dynamic Incorrect Read Disturb Fault* (**dIRF**), where a write operation immediately followed by a read operation does not change the logical value stored in the faulty memory cell, but returns an incorrect output.

Table 1 shows the FPs that model these FFMs in a compact notation using the variables $x,y \in \{0,1\}$ where y=not(x).

2.1.2 Two-cells two-operations dynamic faults

Two-cells two-operations dynamic faults are characterized by #FMC=2 and m=2. In this case, we have to consider 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 cells, i.e., 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 is given in [5]. Only a subset of these FPs has been demonstrated to be realistic [5] [11]. We will focus on this subset only, obtaining the following FFMs:

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

Table 1 shows the FPs related to each FFM.

Table 1: Realistic Dynamic FFM

SINGLE-CELL TWO-OPERATIONS DYNAMIC FAULTS	
FFM	**FPs**
dRDF	$<xw_xr_x/y/y>$ $<xw_yr_y/x/x>$
dDRDF	$<xw_xr_x/y/x>$ $<xw_yr_y/x/y>$
dIRF	$<xw_xr_x/x/y>$ $<xw_yr_y/y/x>$
TWO-CELLS TWO-OPERATIONS DYNAMIC FAULTS	
FFM	**FPs**
dCFds	$<xw_xr_x;x/y/->$ $<xw_xr_x;y/x/- >$ $<xw_yr_y;x/y/- >$ $<xw_yr_y;y/x/- >$
dCFrd	$<x;xw_xr_x/y/y>$ $<y;xw_xr_x/y/y>$ $<x;xw_yr_y/x/x>$ $<y;xw_yr_y/x/x>$
dCFdrd	$<x;xw_xr_x/y/x>$ $<y;xw_xr_x/y/x>$ $<x;xw_yr_y/x/y>$ $<y;xw_yr_y/x/y>$
dCFir	$<x;xw_xr_x/x/y>$ $<y;xw_xr_x/x/y>$ $<x;xw_yr_y/y/x>$ $<y;xw_yr_y/y/x>$

2.2 Linked faults taxonomy

In some cases it is possible that the effect of a FFM influences another functional fault. If these faults share the same *a*-cell and/or *v*-cell, the FFMs are linked; otherwise, they are simple or unlinked. As an example let's consider the Disturb Coupling Faults [5] described by the following two FPs: $FP_1=<0w_1;0/1/->$, and $FP_2=<0w_1;1/0/->$.

Figure 1: Example of Linked Fault

Figure 1 shows a *n*-cells memory affected by the two FPs (FP_1 and FP_2) having different *a*-cells (a_1, a_2) and the same *v*-cell (*v*). *i*, *j* and *k* represent the addresses of a_1, a_2 and *v* respectively, with $i<j<k$. According to FP_1, by performing "$0w_1$" on cell *i*, the *v*-cell *k* flips from 0 to 1; than performing "$0w_1$" (according to FP_2) on cell *j*, the *v*-cell *k* changes its value again, from 1 to 0. The global result is that the fault effect is masked by the application of FP_2, since FP_2 has a fault effect (F) opposite to FP_1.

Looking at the example of Figure 1, we can define that two FPs, $FP_1=<S_1/F_1/R_1>$ and $FP_2=<S_2/F_2/R_2>$, are linked, and denoted by "$FP_1 \rightarrow FP_2$", if both of the following conditions are satisfied:

- FP_2 masks FP_1, i.e., $F_2=not(F_1)$, where F_i is the faulty behavior of FP_i, i.e., the value (state) stored in the *v*-cells after applying S_i (see Section 2);
- The Sensitizing operation (S_2) of FP_2 is applied after S_1, on either the *a*-cell or *v*-cell of FP_1.

To detect Linked Faults (LFs), it is necessary to detect in isolation at least one of the FPs that compose the fault (i.e., without allowing the other FP to mask the fault) [17].

In the sequel, we detail the taxonomy of realistic LFs. The classification is based on the number of memory cells involved by the fault. We consider the set of realistic linked faults proposed in [17]. Each linked fault is described using a compact FP formalism. This notation resorts to the variables *z, j,k,x,y*$\in\{0,1\}$ where: $x=not(y)$ and $k=not(j)$

2.2.1 Realistic single cell linked faults

Single-cell linked faults involve a single memory location where all the FPs are sequentially applied. The set of realistic single cell linked faults, reported in Table 2, has been published and validated in [17].

2.2.2 Realistic two-cells linked faults

Two-cells linked faults involve two distinct memory cells: one *a*-cell, and one *v*-cell. We have two different cases: (i) $a < v$, and (ii) $v < a$. The set of realistic two-cells linked faults published in [17] are reported in Table 2. Realistic two cells LFs can be clustered in three different classes:

- $LF2_{aa}$: LFs that share both the *a*-cell and *v*-cell;
- $LF2_{av}$: LFs where FP_1 is a two cells FP and FP_2 is a single cell FP;
- $LF2_{va}$: LFs where FP_1 is the single cell FP and FP_2 is the two cells FP.

2.2.3 Realistic three-cells linked faults

Three cells linked faults are composed of FPs sharing the same *v*-cells, but having different *a*-cells (a_1 and a_2). Considering the possible mutual positions of a_1, a_2 and *v*, realistic three-cells fault models proposed in [17] belong to the following two situations: $a_1<v<a_2$, and $a_2<v<a_1$. In fact a three-cells linked fault is composed of two two-cells fault models that share at least one cell, therefore realistic three-cells linked faults can be represented by the same fault primitives used to represent two cell LFs (see Table 2). A detailed explanation of the motivations that lead to this conclusion can be found in [17].

For the sake of readability we will not report the full list of three cells linked faults, since they do not introduce any additional fault primitive in the fault list.

Table 2: Realistic Linked FFMs

SINGLE-CELL STATIC LFS			
FFM	**Fps**	**S_1**	**S_2**
TF→WDF	$< S_1 / x / - > \rightarrow < S_2 / y / - >$	xw_y	xw_x
WDF→WDF	$< S_1 / x / - > \rightarrow < S_2 / y / - >$	yw_y	xw_x
DRDF→WDF	$< S_1 / x / y > \rightarrow < S_2 / y / - >$	yr_y	xw_x
TF→RDF	$< S_1 / x / - > \rightarrow < S_2 / y / y >$	xw_y	xr_x
WDF→RDF	$< S_1 / x / - > \rightarrow < S_2 / y / y >$	yw_y	xr_x
DRDF→RDF	$< S_1 / x / - > \rightarrow < S_2 / y / y >$	yr_y	xr_x
TWO-CELLS STATIC $LF2_{aa}$S			
FFM	**Fps**	**S_1**	**S_2**
CFds→CFds	$<S_1 ; x / y /-> \rightarrow <S_2 ; y / x / - >$	kw_j , jw_j, jr_j	jw_k , jw_j , jr_j
CFtr→CFds	$<z ; S_1 / x /-> \rightarrow <S_2 ; x / y / - >$	xw_y	jw_k
CFwd→CFds	$<z ; S_1 / y /-> \rightarrow < S_2 ; y / x / - >$	xw_x	jw_k
CFdr→CFds	$<z ; S_1 / y /x > \rightarrow <S_2 ; y / x / - >$	xr_x	jw_k
CFds→CFwd	$<S_1 ; x / y /- > \rightarrow <z ; S_2 / x / - >$	jw_k , jw_j , jr_j	yw_y
CFtr→CFwd	$<z ; S_1 / x /- > \rightarrow <z ; S_2 / y / - >$	xw_y	xw_x
CFwd→CFwd	$<z ; S_1 / x /- > \rightarrow <z ; S_2 / y / - >$	yw_y	xw_x
CFdr→CFwd	$<z ; S_1 / x /y > \rightarrow <z ; S_2 / y / - >$	yr_y	xw_x
CFds→CFrd	$<S_1 ; x / y /- > \rightarrow <z ; S_2 / x / x >$	jw_k , jw_j , jr_j	yr_y
CFtr→CFrd	$<z ; S_1 / y /- > \rightarrow <z ; S_2 / x / x >$	yw_x	yr_y
CFwd→CFrd	$<z ; S_1 / y /- > \rightarrow <z ; S_2 / x / x >$	xw_x	yr_y
CFdr→CFrd	$<z ; S_1 / y /x > \rightarrow <z ; S_2 / x /x >$	xr_x	yr_y
TWO-CELLS STATIC $LF2_{av}$S			
FFM	**I , s**	**S_1**	**S_2**
CFds→WDF	$<S_1 ; x / y /-> \rightarrow <S_2 / x / - >$	jw_k , jw_j , jr_j	yw_y
CFtr→ WDF	$<z ; S_1 / y /-> \rightarrow <S_2 / x / - >$	yw_x	yw_y
CFwd→WDF	$<z ; S_1 / y /-> \rightarrow < S_2 / x / - >$	xw_x	yw_y
CFdr→WDF	$<z ; S_1 / y /x > \rightarrow <S_2 / x / - >$	xr_x	yw_y
CFds→RDF	$<S_1 , x / y /- > \rightarrow <S_2 / x / x >$	jw_k , jw_j , jr_j	yr_y
CFtr→RDF	$<z ; S_1 / y /- > \rightarrow <S_2 / x / x >$	yw_x	yr_y
CFwd→RDF	$<z ; S_1 / y /- > \rightarrow <S_2 / x / x >$	xw_x	yr_y
CFdr→RDF	$<z ; S_1 / y /x > \rightarrow <S_2 / x / x >$	xr_x	yr_y
TWO-CELLS STATIC $LF2_{va}$S			
FFM	**Fps**	**S_1**	**S_2**
'DF→CFds	$<S_1 / x /-> \rightarrow <S_2 ; x / y / - >$	yw_y	jw_k , jw_j , jr_j
TF→CFds	$<S_1 / x /-> \rightarrow <S_2 ; x / y / - >$	xw_y	jw_k , jw_j , jr_j
DRDF→CFds	$<S_1 / x /y > \rightarrow <S_2 ; x / y / - >$	yr_y	jw_k , jw_j , jr_j
WDF→CFwd	$<S_1 / x /-> \rightarrow <z ; S_2 / y / - >$	yw_y	xw_x
TF→CFwd	$<S_1 / x /-> \rightarrow <z ; S_2 / y / - >$	xw_y	xw_x
DRDF→CFwd	$<S_1 / x /y > \rightarrow <z ; S_2 / y / - >$	yr_y	xw_x
WDF→CFrd	$<S_1 / x /-> \rightarrow <z ; S_2 / y / y >$	yw_y	xr_x
TF→CFrd	$<S_1 / x /-> \rightarrow <z ; S_2 / y / y >$	xw_y	xr_x
DRDF→CFrd	S1 / x /y $> \rightarrow <z ; S_2 / y / y >$	yr_y	xr_x

3 March AB

Figure 2 introduces March AB, whose complexity is 22*n*. March AB is able to detect the whole set of realistic static linked faults and dynamic faults proposed in Section 2. Moreover it is able to deal with the full set of static unlinked faults published in [5]. With this algorithm we reduce the test complexity by 4*n* w.r.t. March RAW [11], the only published march test having the same fault coverage, with a consequent reduction of the test time of about 15.4%.

March AB has been designed by resorting to the automatic march test generation algorithm introduced in [19]. Moreover, using the same generation algorithm we defined the set of *Fault Coverage Conditions* (FCC) needed to detect the target faults. Each FCC specifies the March Elements (MEs) able to detect the target fault. The FCCs for the set of faults listed in Section 2 will be introduced in Section 4 and will be used to prove that our March AB actually covers the given list of faults.

Figure 2: March AB, with complexity of 22*n*

4 March AB validation

The fault coverage conditions can be directly derived from the functional fault models definitions and in particular from the fault primitives composing each FFM.

FCCs are expressed using the march test notation by adding the following symbols:

- '…': any operation;
- 'OP(*d*)': any operation 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;
- '$\{M_1M_2\}$': the MEs (M_i) included in braches must be executed in that order;
- '[…]': includes one *relation* between two or more MEs. A relation is specified by logical operators between MEs (AND, OR). As an example, the coverage condition '[$\{M_1\ M_2\}$ OR $\{M_3\ M_4\}$] AND M_5' means that the march test can be either $\{M_1\ M_2\ M_5\}$ or $\{M_3\ M_4\ M_5\}$.

In the following subsections we introduce and analyze the fault coverage conditions able to cover the whole set of fault primitives introduced in Section 3. A short proof about the coverage is given for each coverage condition.

4.1 Dynamic faults FCCs

The FCCs for single-cell dynamic faults are:

- **DFCC1**: $[\{\Updownarrow((\ldots)^*,OP(x))\Updownarrow(w_x,r_x,(\ldots)^*)\}OR\{\Updownarrow((\ldots)^*,OP(x),w_x,r_x,(\ldots)^*)\}]$
- **DFCC2**: $[\{\Updownarrow((\ldots)^*,OP(x))\Updownarrow(w_y,r_y,(\ldots)^*)\}OR\{\Updownarrow((\ldots)^*,OP(x),w_y,r_y,(\ldots)^*)\}]$
- **DFCC3**: $[\{\Updownarrow((\ldots)^*,OP(x))\Updownarrow(w_x,r_x,r_x,(\ldots)^*)\}OR\{\Updownarrow((\ldots)^*,OP(x))\Updownarrow(w_x,r_x)\Updownarrow(r_x,(\ldots)^*)\}OR$ $\{\Updownarrow((\ldots)^*,OP(x),w_x,r_x,r_x,(\ldots)^*)\}OR\{\Updownarrow((\ldots)^*,OP(x),w_x,r_x)\Updownarrow(r_x(\ldots)^*)\}]$
- **DFCC4**: $[\{\Updownarrow((\ldots)^*,OP(x))\Updownarrow((w_y,r_y)+,r_y,(\ldots)^*)\}OR$ $\{\Updownarrow((\ldots)^*,OP(x))\Updownarrow((w_y,r_y)+)\Updownarrow(r_y,(\ldots)^*)\}OR\{\Updownarrow((\ldots)^*,OP(x),(w_y,r_y)^+,r_y,(\ldots)^*)\}OR$ $\{\Updownarrow((\ldots)^*,OP(x),(w_y,r_y)+)\Updownarrow(r_y,(\ldots)^*)\}]$

DFCC1 detects FP=<$xw_xr_x/y/y$> (dRDF see Table 1) since the OP(x) initializes the memory cell and the w_x,r_x sensitizes the fault effect. Finally the r_x detects the fault effect (it returns the wrong value y in case of fault). Note that the initializing, sensitizing and detecting operations can belong to the same ME.

DFCC2 detects FP=<$xw_yr_y/x/x$> (dRDF see Table 1). The proof of its coverage is the same of DFCC1.

DFCC3 detects FP=<$xw_xr_x/y/x$> (dDRDF see Table 1). This type of error requires an additional read to detect the fault effect. The sequence w_x,r_x sensitizes the fault and the read operation returns the correct value (the fault is not detected). Adding an additional r_x the fault is detected. As for DFCC1, the initializing, sensitizing and detecting operations can either belong to the same ME or not. DFCC3 includes all the possible combinations.

DFCC4 detects FP=<$xw_yr_y/x/y$> (dRDF see Table 1), it works in the same way as DFCC3. The only difference is that after the initialization (OP(x)), it is possible to repeat one or more times the sensitizing operations ($(w_y,r_y)+$), before the detecting operation (r_y). As an example, in the march test '$\{\Updownarrow(OP(x))\ \Updownarrow(w_y,r_y,w_y,r_y,r_y)\}$', where the w_y,r_y is repeated two times, after the execution of the first group w_y,r_y the fault is sensitized and the memory cell is set in the wrong state x. The next group w_y,r_y is therefore applied on a memory cell in a state x, hence the fault is again sensitized and the memory still reach the state x. Finally the last r_y detects the fault effect.

Concerning the remaining FPs, FP=<$xw_xr_x/x/y$> (dIRF see Table 1) can be detected by either DFCC1 or DFCC3 and FP=<$xw_yr_y/y/x$> (dIRF see Table 1) can be detected by either DFCC2 or DFCC4.

Table 3 reports for each FFM (column 1) the related set of FPs (column 2), and for each FP the related DFFCs (column 3). Note that DFCC3 and DFCC4 include respectively DFCC1 and DFCC2. That means that either DFCC3 or DFCC4 guarantees the detection of every single-cell dynamic faults (detailed in Section 2.1.1).

Table 3: Condition for detecting single-cell dynamic fault

FFM	FP	FCC
dRDF	$<xw_xr_x/y/y>$	DFFC1 or DFCC3
	$<xw_yr_y/x/x>$	DFFC2 or DFCC4
dDRDF	$<xw_xr_x/y/x>$	DFFC3
	$<xw_yr_y/x/y>$	DFFC4
dIRF	$<xw_xr_x/x/y>$	DFFC1 or DFFC3
	$<xw_yr_y/y/x>$	DFFC2 or DFFC4

The FCCs for two-cells dynamic faults are:

- **DFCC5**: $[\{⇧(r_x,(\ldots,OP(x))^*,w_x,r_x,(\ldots)^*)\}AND\{⇩(r_x,(\ldots,OP(x))^*,w_x,r_x,(\ldots)^*)\}]OR$
  $[\{⇧((\ldots)^*,OP(x),w_x,r_x)⇧(r_x,(\ldots)^*)\}AND\{⇩((\ldots)^*,OP(x),(w_x,r_x)\ ⇧(r_x,(\ldots)^*)\}]$
- **DFCC6**: $[\{⇧(r_y,(\ldots)^*,OP(x),w_x,r_x,(\ldots)^*)\}AND\{⇩(r_y,(\ldots)^*,OP(x),w_x,r_x,(\ldots)^*)\}]OR$
  $[\{⇧((\ldots)^*,OP(x),w_x,r_x,(\ldots)^*,OP(y))⇧(r_y,(\ldots)^*)\}AND$
  $\{⇩((\ldots)^*,OP(x),w_x,r_x,(\ldots)^*,OP(y))\ ⇧(r_y,(\ldots)^*)\}]$
- **DFCC7**: $[\{⇧(r_x,(\ldots,OP(x))^*,w_y,r_y,(\ldots)^*)\}AND\{⇩(r_x,(\ldots,OP(x))^*,w_y,r_y,\ (\ldots)^*)\}]OR$
  $[\{⇧((\ldots,OP(x))^*,w_y,r_y,(\ldots,OP(x))^*)⇧(r_x,(\ldots)^*)AND$
  $\{⇩((\ldots,OP(x))^*,w_y,r_y,(\ldots,OP(x))^*)⇧(r_x,(\ldots)^*)\}]$
- **DFCC8**: $[\{⇧(r_x,(\ldots,OP(x))^*,w_y,r_y,(...,OP(y))^*)⇧(r_y,(\ldots)^*)\}AND$
  $\{⇩(r_x,(\ldots,OP(x))^*,w_y,r_y,(...,OP(y))^*)\ ⇩(r_y,(\ldots)^*)\}]OR$
  $[\{⇧((\ldots)^*,OP(x),w_y,r_y,(...,OP(y))^*)$
  $⇧(r_y,(\ldots)^*,)\}AND\{⇩((\ldots)^*,OP(x),w_y,r_y,(...,OP(y))^*)\ ⇧(r_y,(\ldots)^*)\}]$
- **DFCC9**: $[\{⇳((\ldots)^*,OP(y))⇧((\ldots,OP(x))+,w_x,r_x,(...,OP(x))^*)\}AND$
  $\{⇳((\ldots)^*,OP(y))⇩((\ldots,OP(x))+,w_x,r_x,(...,OP(x))^*)\}]$
- **DFCC10**: $[\{⇳((\ldots)^*,OP(x))⇧((\ldots,OP(x))^*,w_y,r_y,(\ldots,OP(y))^*)\}AND$
  $\{⇳((\ldots)^*,OP(x))⇩((\ldots,OP(x))^*,w_y,r_y,(\ldots,OP(y))^*)\}]$
- **DFCC11**: $[\{⇳((\ldots)^*,OP(y))⇧((\ldots,OP(x))+,w_x,r_x)⇳(r_x,(\ldots)^*)\}AND$
  $\{⇳((\ldots)^*,OP(y))⇩((\ldots,OP(x))+,w_x,r_x)⇳(r_x,(\ldots)^*)]$
- **DFCC12**: $[\{⇳((\ldots)^*,OP(x))⇧((\ldots,OP(x))^*,(w_y,r_y)+)⇳(r_y,(\ldots)^*)\}AND$
  $\{⇳((\ldots)^*,OP(x))⇩((\ldots,OP(x))^*,(w_y,r_y)+)⇳(r_y,(\ldots)^*)\}]$

The coverage conditions for two-cells dynamic faults are more complex w.r.t those for single-cells. The main reason is that the relation between aggressor and victim cells has to be taken into account. As introduced in Section 2.1.2, 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.

DFCC5 detects FP=<$xw_xr_x;x/y/-$> (dCFds, see Table 1). The left term of the '*OR*' operand includes both sensitizing and detecting operations in the same ME, whereas the right term includes sensitizing and detecting operations in two distinct MEs. The ME '⇧(r_x,(…,OP(x))*,w_x,r_x,(…)*)' first performs the operations on the *a*-cell ($a<v$). Since the *v*-cell is set to '*x*' the sequence w_x,r_x, sensitizes the fault effect. Moving on the *v*-cell, the first read (r_x) of the ME detects the fault (it reads '*y*' instead of '*x*' in case of fault). Changing the address order allows to cover the case $v<a$.

The MEs '⇧((…)*,OP(x),w_x,r_x) ⇳(r_x,(…)*)' first perform the operations on the *v*-cell, setting its value to '*x*'. Then it moves on the *a*-cell where the sequence w_x,r_x sensitizes the fault (*v*-cell is set to '*y*' in case of fault). The following ME begins with a read operation (r_x) that detects the fault. This ME detects the fault when $v<a$. Inverting the address order it also detects the case of a<v.

In the same way DFCC6, DFCC7 and DFCC8 cover the remaining dCFds (<$xw_xr_x;y/x/-$>, <$xw_yr_y;x/y/-$> and <$xw_yr_y;y/x/-$>, see Table 1).

DFCC9 detects FP_1=<$x;xw_xr_x/y/y$> and FP_2=<$y;xw_xr_x/y/y$> (dCFrd, see Table 1). The ME '⇧((…,OP(x))*,w_x,r_x,(...,OP(x))*)' sensitizes and detects FP_2 when v<a. Memory cells are initialized to '*y*' by the previous ME '(⇳((…)*,OP(y)))'. The operations are first performed on the *v*-cell, so w_x,r_x sensitizes the fault and the r_x detects it. The same ME guarantees the detection of FP_1 when $a<v$. It performs the operations on the *a*-cell setting its value to '*x*'. Then it moves on the *v*-cell where the sequence w_x,r_x sensitizes the fault and the r_x also detects it. Changing the address order ensures the detection of both FP_2 (when a<v) and FP_1 (when v<a).

In the same way DFCC10 covers the remaining dCFrd (<$x;xw_yr_y/x/x$> and <$y;xw_yr_y/x/x$>, see Table 1).

DFCC11 detects FP_1=<$x;xw_xr_x/y/x$> and FP_2=<$y;xw_xr_x/y/x$> (dCFdrd, see Table 1). The ME '⇧((…,OP(x))*,w_x,r_x)' sensitizes FP_2 when $v<a$. Memory cells are initialized to '*y*' by the previous ME '⇧((…)*,OP(y))'. The operations are first performed on the *v*-cell, so w_x,r_x sensitizes the fault and the r_x performed by the following ME (⇳(r_x,(…)*)) detects it. The same ME guarantees the detection of FP_1 when a<v. It performs the operations on the *a*-cell setting its value to 'x'. Then it

moves on the *v*-cell and the sequence w_x,r_x sensitizes the fault and the r_x of the next ME detects it. Changing the address order ensures the detection on both FP_2 (when a<v) and FP_1 (when v<a).

In the same way DFCC12 covers the remaining dCFdrd (<x;x$w_y r_y$/x/y> and <y;x$w_y r_y$/x/y>, see Table 1).

In a similar way, dCFir (see Table 1) is composed of FP_1=<x;x$w_x r_x$/x/y>, FP_2=<y;x$w_x r_x$/x/y>, FP_3=<x;x$w_y r_y$/y/x> and FP_4=<y;x$w_y r_y$/y/x>. FP_1 and FP_2 are detected by DFFC9 or DFFC11. FP_3 and FP_4 are detected by DFCC10 or DFCC12.

Table 4 reports for each fault model (column 1) its set of FPs (column 2), and for each FP the related fault coverage conditions (column 3).

Table 4: Condition for detecting two-cell dynamic faults

FFM	FP	DFFC
dCFds	<x$w_x r_x$;x/y/->	DFFC5
	<x$w_x r_x$;y/x/->	DFFC6
	<x$w_y r_y$;x/y/->	DFFC7
	<x$w_y r_y$;y/x/->	DFFC8
dCFrd	<x;x$w_x r_x$/y/y>	DFFC9
	<y;x$w_x r_x$/y/y>	
	<x;x$w_y r_y$/x/x>	DFFC10
	<y;x$w_y r_y$/x/x>	
dCFdrd	<x;x$w_x r_x$/y/x>	DFFC11
	<y;x$w_x r_x$/y/x>	
	<x;x$w_y r_y$/x/y>	DFFC12
	<y;x$w_y r_y$/x/y>	
dCFir	<x;x$w_x r_x$/x/y>	DFFC9 or DFFC11
	<y;x$w_x r_x$/x/y>	
	<x;x$w_y r_y$/y/x>	DFCC10 or DFCC12
	<y;x$w_y r_y$/y/x>	

4.2 Linked faults FCCs

As described in Section 2.2.1, there are two main classes of single-cell LFs having the same FP_2. The first one has FP_2={WDF} and FP_1={TF, WDF} (see Table 2). In this class, FP_1 is sensitized by S_1={xw_y, yw_y,} and FP_2 by S_2={xw_x}. The ME '⇳(r_y,w_x,r_x,w_x,r_x)' detects WDF in isolation by applying the fourth operation (w_x) that sensitizes the fault, then the read (r_x) observes the fault effect. FP_1 cannot be sensitized since the ME doesn't contain any operations belonging to S1. When FP_1={DRDF}, S_1={yr_y} the MEs '⇳(r_x,w_y,r_y,w_y,r_y) ⇳(r_y,…)' cover FP_1 in isolation. The last read (r_y) of the first ME sensitizes the fault when the second ME detects it.

The second class has FP_2={RDF} and FP_1={TF, WDF, DRDF} (see Table 2). FP_2 is sensitized by S_2={xr_x} and FP_1 by S_1={xw_y, yw_y, yr_y}. Therefore the ME '⇳(r_x,w_y,r_y,w_y,r_y)' detects RDF in

isolation by the first operation (r_x). FP_1 is sensitized after FP_2 therefore masking cannot occur. The LFCC covering the entire set of single-cell LFs is:

- **LFCC1**: $\Updownarrow(r_x,w_y,r_y,w_y,r_y)\ \Updownarrow(r_y,w_x,r_x,w_x,r_x)$

The two cells LFs detection is more complex than those for single-cell faults, because the relations between aggressor and victim address constraints ($a<v$ and $v<a$) must be respected. Referring to the two cells LFs classification (Section 0), we order LFs having the same FP_2. In the first group of faults, where FP_1=FP_2={CFds} (see Table 2). FP_2 is sensitized by $S_2=\{jw_k\ ,\ jw_j\ ,\ jr_j\}$. We investigate each sequence of operations belong to S_2, and the relative LF.

The first instance $S_2=\{jw_k\}$ implies LF='$<S_1;x/y/->\rightarrow<\ jw_k;y/x/->$', where $S_1=\{kw_j,\ jw_j,\ jr_j\}$, $j,k,x,y=\{0,1\}$, y=not(x), and j=not(k).

Setting the values $j=y$, $k=x$, $S_1=\{xw_y,\ yw_y,\ yr_y\}$ and $S_2=\{yw_x\}$, the ME '$\Downarrow(r_y,w_x,r_x,w_x,r_x)$' can detect FP_2 in isolation when $a>v$. In this case the first accessed cell is the a-cell, and only FP_2 can be sensitized since the v-cell is in the y state. Therefore the second operation (w_x) sensitizes the fault FP_2 and no other faults can be sensitized, so when v-cell is accessed, the first read (r_y) detects the fault. In the same way, to cover the case $a<v$ the ME '$\Uparrow(r_y,w_x,r_x,w_x,r_x)$' is required.

Setting the opposite values $j=x$, $k=y$, $S_1=\{yw_x,\ xw_x,\ xr_x\}$ and $S_2=\{xw_y\}$, the MEs '$\Downarrow(r_x,w_y,r_y,w_y,r_y)\ \Updownarrow(r_{y,\ldots})$' can detect FP_2 in isolation when $a<v$. The v-cell is first accessed setting its value to the y state (w_y). Then the a-cell is accessed and the second operation (w_y) sensitizes the fault FP_2 in isolation. The first read (r_y) on the second ME detects the fault effect. In the same way, to cover the case $a>v$ the MEs '$\Uparrow(r_x,w_y,r_y,w_y,r_y)\ \Updownarrow(r_{y,\ldots})$' are required. No other faults can be sensitized, since the MEs don't include the required operations.

The second instance $S2=\{jr_j\}$ implies LF='$<S_1;x/y/->\rightarrow<\ jrj;y/x/->$', where $S_1=\{kw_j\ ,\ jw_j,\ jr_j\}$, $j,k,x,y=\{0,1\}$, y=not(x), and j=not(k).

Setting the values $j=y$, $k=x$, $S_1=\{xw_y,\ yw_y,\ yr_y\}$ and $S_2=\{yr_y\}$, the ME '$\Downarrow(r_y,w_x,r_x,w_x,r_x)$' can detect FP_2 in isolation when $a>v$. Accessing the a-cell, the first operation (r_y) sensitizes the fault FP_2 in isolation. When the v-cell is accessed, the first read (r_y) detects the fault. In the same way, to cover the case $a<v$ the ME '$\Uparrow(r_y,w_x,r_x,w_x,r_x)$' is required.

Setting the opposite values $j=x$, $k=y$, $S_1=\{yw_x,\ xw_x,\ xr_x\}$, and $S_2=\{xr_x\}$, the MEs '$\Downarrow(r_x,w_y,r_y,w_y,r_y)\ \Updownarrow(r_{y,\ldots})$' can detect FP_2 in isolation when $a<v$. The v-cell is first accessed, setting its state to the y state (w_y). Then the a-cell is accessed and the first operation (r_x) sensitizes the fault FP_2 in isolation.

The first read operation on the second ME detects the fault effect. In the same way, to cover the case $a>v$ the MEs '⇧(r_x,w_y,r_y,w_y,r_y) ⇳($r_{y,...}$)' are required.

The last instance S_2={jw_j} implies LF='<S_1;x/y/->→< jw_j;y/x/->', where S_1={kw_j, jw_j, jr_j}, j,k,x,y={0,1}, y=not(x), and j=not(k).

Setting the values $j=y$, $k=x$, S_1={xw_y, yw_y, yr_y} and S2={$_{ywy}$}, the MEs '⇩(r_x,w_y,r_y,w_y,r_y) ⇳($r_{y,...}$)' can detect FP_2 in isolation when $a<v$. The v-cell is first accessed setting its state to the value y (w_y). Then the a-cell is accessed and the fourth operation (w_y) sensitizes the fault FP_2 in isolation. The first read operation on the second march test detects the fault effect. When S_1={yr_y}, the last operation (r_y) will mask the fault. In order to avoid this condition, the MEs are refined as '⇩(r_x,w_y,r_y,w_y,r_y) ⇩(r_y,w_x,r_x,w_x,r_x) ⇳($r_{x,...}$)', where FP_1 is sensitized by the first operation of the second ME and observed by the third ME. In the same way, to cover the case $a>v$ the MEs '⇧(r_x,w_y,r_y,w_y,r_y) ⇧(r_y,w_x,r_x,w_x,r_x) ⇳($r_{x,...}$)' are required.

Setting the opposite values $j=x$, $k=y$, S_1={yw_x, xw_x, xr_x}, and S_2={xw_x}, the ME '⇩(r_y,w_x,r_x,w_x,r_x)' can detect FP_2 in isolation when $a>v$. The a-cell is first accessed and the fourth operation (w_x) sensitizes the fault FP_2 in isolation. The first read operation on the v-cell will detect the fault effect. When S_1={xr_x}, the last operation (r_y) will mask the fault. Therefore it requires the additional ME '⇩(r_x,w_y,r_y,w_y,r_y)' in order to sensitize and detect in isolation FP_1. In the same way, to cover the case $a<v$ the MEs '⇧(r_y,w_x,r_x,w_x,r_x) ⇧(r_x, w_y,r_y,w_y,r_y)' are required.

Finally, the full set of CFds linked to CFds is detected by the following coverage condition:

- **LFCC2**: ⇩(r_x,w_y,r_y,w_y,r_y) ⇩(r_y,w_x,r_x,w_x,r_x) ⇧(r_x,w_y,r_y,w_y,r_y) ⇧(r_y,w_x,r_x,w_x,r_x) ⇳ ($r_{x,...}$)

LFCC2 is still valid when FP_1 is a CFtr, a CFwd, or a CFdr, since each CFds is detected in isolation. Similarly, LFCC2 is also valid for the remaining LF2aa, LF2av and LF2va (see Table 2).

Three cells LFs are composed of two cells FPs (see 2.2.3) sharing the same v-cells but having different a-cells (a_1 and a_2). [17] proves that the conditions to detect two cells LFs are enough to detect all the three cells LFs. Therefore, LFCC2 ensures the detection of the entire three-cells LFs space.

4.3 March test validation

In order to validate March AB we have to prove that it includes all the DFCCs and LFCCs introduced in Section 4.1 and 4.2. In 4.2 we already proved that LFCC2 includes LFCC1. In this section we prove that LFCC2 also includes all the DFCCs. Table 5 shows which part (MEs) of LFCC2 covers

each DFCC. Column 1 reports the coverage condition number, column 2 the relevant condition part, whereas columns 3 shows the MEs containing the rules.

Table 5: DFCCs Evidence in LFCCs

# DFCC	DFCC	LFCC2 MEs
DFCC3	⇳((...)*,OP(x), w_x,r_x) ⇳(r_x (…)*)	⇩(r_y,w_x,r_x,w_x,r_x) ⇧(r_x,w_y,r_y,w_y,r_y)
DFCC3	⇳((...)*,OP(x), w_x,r_x) ⇳(r_x (…)*)	⇧(r_y,w_x,r_x,w_x,r_x) ⇳(r_x,...)
DFCC4	⇳((...)*,OP(x), (w_y,r_y)+) ⇳(r_y,(…)*)	⇩(r_x,w_y,r_y,w_y,r_y) ⇩(r_y,w_x,r_x,w_x,r_x)
DFCC4	⇳((...)*,OP(x), (w_y,r_y)+) ⇳(r_y,(…)*)	⇧(r_x,w_y,r_y,w_y,r_y) ⇧(r_y,w_x,r_x,w_x,r_x)
DFCC5	⇧((…)*,OP(x),w_x,r_x) ⇧(r_x,(…)*) and ⇩((…)*,OP(x),(w_x,r_x) ⇩(r_x,(…)*)	⇧(r_y,w_x,r_x,w_x,r_x) ⇳(r_x,...) and ⇩(r_y,w_x,r_x,w_x,r_x) ⇧(r_x,w_y,r_y,w_y,r_y)
DFCC6	⇧(r_y,(…)*,OP(x),w_x,r_x,(…)*) and ⇩(r_y,(…)*,OP(x),w_x,r_x,(…)*)	⇩(r_y,w_x,r_x,w_x,r_x) and ⇧(r_y,w_x,r_x,w_x,r_x)
DFCC7	⇧(r_x,(…,OP(x))*,w_y,r_y,(…)*) and ⇩(r_x,(… ,OP(x))*,w_y,r_y, (…)*)	⇩(r_x,w_y,r_y,w_y,r_y) and ⇧(r_x,w_y,r_y,w_y,r_y)
DFCC8	⇧(r_x,(…,OP(x))*,w_y,r_y,(...,OP(y))*) ⇧(r_y,(…)*) and ⇩(rx,(…,OP(x))*,w_y,r_y,(...,OP(y))*) ⇩(r_y,(…)*)	⇩(r_x,w_y,r_y,w_y,r_y) ⇩(r_y,w_x,r_x,w_x,r_x)and ⇧(r_x,w_y,r_y,w_y,r_y) ⇧(r_y,w_x,r_x,w_x,r_x)
DFCC9	⇳ ((…)*,OP(y)) ⇧((…,OP(x))+,w_x,r_x,(...,OP(x))*) and ⇳((…)*,OP(y)) ⇩((…,OP(x))+,w_x,r_x,(...,OP(x))*)	⇩(r_x,w_y,r_y,w_y,r_y) ⇩(r_y,w_x,r_x,w_x,r_x)and ⇧(r_x,w_y,r_y,w_y,r_y) ⇧(r_y,w_x,r_x,w_x,r_x)
DFCC10	⇳((…)*,OP(x)) ⇧((…,OP(x))*,w_y,r_y,(…,OP(y))*) and ⇳((…)*,OP(x)) ⇩((…,OP(x))*,w_y,r_y,(…,OP(y))*)	⇩(r_y,w_x,r_x,w_x,r_x) ⇧(r_x,w_y,r_y,w_y,r_y) and ⇩(r_x,w_y,r_y,w_y,r_y)[1]
DFCC11	⇳((…)*,OP(y)) ⇧((…,OP(x))+,w_x,r_x) ⇳(r_x,(…)*) and ⇳((…)*,OP(y)) ⇩((…,OP(x))+,w_x,r_x) ⇳(r_x,(…)*)	⇩(r_y,w_x,r_x,w_x,r_x) ⇧(r_x,w_y,r_y,w_y,r_y)[1] and ⇧ (r_y,w_x,r_x,w_x,r_x) ⇳ (r_x,...)[1]
DFCC12	⇳((…)*,OP(x)) ⇧((…,OP(x))*,(w_y,r_y)+) ⇳(r_y,(…)*) and ⇳ ((…)*,OP(x)) ⇩((…,OP(x))*,(w_y,r_y)+) ⇳(r_y,(…)*)	⇩(r_x,w_y,r_y,w_y,r_y) ⇩(r_y,w_x,r_x,w_x,r_x)[1] and ⇧(r_x,w_y,r_y,w_y,r_y) ⇧(r_y,w_x,r_x,w_x,r_x)[1]

Analyzing LFCCs it is important to note that LFCC1 (Section 4.2) are included in LFCC2. In other words LFCC2 also covers single cell LFs.

We can expand LFCC2 exploiting the whole set of values assumed by *x* and *y*. Table 6 shows each ME obtained by setting the *x* and *y* values. Looking at the results we can note that some MEs are redundant. In particular M1=M6, M2=M7, M3=M8 and M4=M9. Note that M10 is included in M4. After removing all the redundancies we obtain five MEs reported in Table 7. Now, LFCC2 exactly correspond to March AB (Figure 2).

[1] For sake of readability the ME that initializes the memory is omitted see Section 4 for details

Table 6: expanded LFCC2

#	MEs	*x*,*y*
M1	⇩(r_0,w_1,r_1,w_1,r_1)	$x = 0, y = 1$
M2	⇩(r_1,w_0,r_0,w_0,r_0)	$x = 0, y = 1$
M3	⇧(r_0,w_1,r_1,w_1,r_1)	$x = 0, y = 1$
M4	⇧(r_1,w_0,r_0,w_0,r_0)	$x = 0, y = 1$
M5	⇳ ($r_{0,...}$)	$x = 0, y = 1$
M6	⇩(r_0,w_1,r_1,w_1,r_1)	$x = 1, y = 0$
M7	⇩(r_1,w_0,r_0,w_0,r_0)	$x = 1, y = 0$
M8	⇧(r_1,w_0,r_0,w_0,r_0)	$x = 1, y = 0$
M9	⇧(r_0,w_1,r_1,w_1,r_1)	$x = 1, y = 0$
M10	⇳ ($r_{1,...}$)	$x = 1, y = 0$

Table 7: reduced LFCC2

#	MEs
M1	⇩(r_0,w_1,r_1,w_1,r_1)
M2	⇩(r_1,w_0,r_0,w_0,r_0)
M3	⇧(r_0,w_1,r_1,w_1,r_1)
M4	⇧(r_1,w_0,r_0,w_0,r_0)
M5	⇳ ($r_{0,...}$)

5 Comparing march tests and simulation results

In this section we compare March AB to already published march tests. The aim of this comparison is to show the effectiveness of March AB. The comparison is performed considering different fault lists. The results will show that March AB is a very interesting solution since it offers the largest coverage with the shortest length. Each march test has been simulated by using the memory fault simulator presented in [20].

The first experiments compare March AB with march tests designed for either static or dynamic faults: March C- [12], March RAW, March RAW1 and March SS [11] .

Table 8: comparison of march tests for dynamic faults

MT	*Complexity*	*Single-cell*			*Two-cell*			
		dRDF	*dDRDF*	*dIRF*	*dCFds*	*dCFrd*	*dCFrdr*	*dCFir*
C-	10*n*	0.39%	0.39%	0.39%	0.83%	0.39%	0.39%	0.39%
RAW1	13*n*	100%	100%	100%	0%	50%	50%	50%
SS	22*n*	25%	0.39%	25%	50%	50%	0.39%	50%
AB	22*n*	100%	100%	100%	100%	100%	100%	100%
RAW	26*n*	100%	100%	100%	100%	100%	100%	100%

Table 8 summarizes the fault coverage for each of the proposed march test and for each of the considered dynamic faults (see Section 2.1). March RAW [11] is explicitly designed to cover the same set of dynamic faults targeted by March AB. As reported in Table 8, March AB offers the same

fault coverage of March RAW but it reduces the complexity of 15.4% (4 operations). We also consider March C- since in [12] the authors demonstrate that, by using an address order customized on the physical layout of the memory it is able to detect dynamic faults. Nevertheless, in our experiments, we considered the memory under test from a functional point of view, without any layout information and we tried to show that without this information the coverage of March C- becomes very marginal: 0.39% (see Table 8).

Of particular interest is the comparison with March SS [11], originally designed to cover the full set of memory static faults published in [5]. As shown in Table 8, March AB has the same complexity of March SS. Even if not reported in the table, March AB is able to cover the same set of static faults of March SS and in addition it covers dynamic faults.

We also compared March AB with the state-of-the-art march tests able to detect linked faults: March LR [14], March A, March B [10], March LA [13], March MSL [18], March RAW [11], March SL [16], and [15].

In particular, we compared March AB with March SL [16] and March MSL [18] since they target the same set of linked faults. The others march test considered in the comparison (A, B, LR, LA and [15]) are still able to detect linked faults, but only a reduced set of fault models w.r.t. the previous ones.

Table 9 summarizes the simulation results in terms of fault coverage for each march test and the related complexity. It targets single cell LFs, two cells LFs and three cells LFs. Comparison results show that the March AB provides the same fault coverage of the March MSL and March SL the two state of the art march tests for linked faults, but it reduces the test complexity of respectively 44.18% and 4.34%.

Table 9: Simulation Results linked faults

MT	***Complexity***	***Single cell LF***	***(Two/Three)-cells***			
			$LF2_{aa}$	$LF2_{av}$	$LF2_{va}$	***All***
LR	14*n*	75%	82%	75%	80%	80%
A	15*n*	66%	75%	60%	73%	69%
B	17*n*	75%	70%	64%	73%	70%
LA	22*n*	83%	87%	83%	86%	86%
AB	22*n*	100%	100%	100%	100%	100%
MSL	23*n*	100%	100%	100%	100%	100%
RAW	26*n*	100%	100%	100%	100%	100%
SL	41*n*	100%	100%	100%	100%	100%
[15]	43*n*	83%	84%	83%	86%	84%

6 Conclusions

This paper proposed March AB, a new march test targeting both static linked faults and dynamic unlinked faults. A detailed description of the coverage conditions needed to detect each fault has been proposed, and the correctness of Marc AB has been proved by demonstrating that it satisfies all the coverage conditions.

Moreover we compared March AB with state-of-the-art algorithms resorting to fault simulation experiments, showing that our test provides the maximum coverage while reducing the test time. March AB allows having a single march test addressing an extended set of faults. 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.

March AB has been design by considering a pure functional model of the memory under test. Future optimization can derive from considerations on the actual layout of the memory that can possibly allow choosing appropriate address orders and data patterns able to reduce the final test lengths.

7 References

[1] Marinissen, E. J.; Prince, B.; Keltel-Schulz D.; Zorian Y.: "Challenges in embedded memory design and test", Design, Automation and Test in Europe, Munich, Germany, March 2005, pp. 722-727 Vol. 2.

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

[3] Hamdioui, S.; Wadsworth, R.; Reyes, J.D.; Van de Goor, A.J.:"Importance of Dynamic Faults for new SRAM Technologies", 8th IEEE European Test Workshop, Maastricht, The Netherlands, May 2003, pp. 29-34.

[4] Al-Ars, Z.; Van de Goor, A.J.: "Static and Dynamic Behavior of Memory Cell Array Opens and Shorts in Embedded DRAMs", IEEE Design Automation and Test in Europe, Munich, Germany, March 2001, pp. 496-503.

[5] Van de Goor, A.J.; Al-Ars, Z. :"Functional Memory Faults: A Formal Notation and a Taxonomy", 18th IEEE VLSI Test Symposium, Marina Del Rey, CA, USA, May 2000, pp.281-289.

[6] Al-Ars, Z.; Van de Goor, A.J. :"Approximating Infinite Dynamic Behavior for DRAM Cell Defects", 20th IEEE VLSI Test Symposium, Monterey, CA, USA, May 2002, pp.401-406.

[7] Dekker, R.; Beenker, F.; Thijssen, L.: "A Realistic Fault Model and Test Algorithms for Static Random Access Memory", IEEE Transaction on CAD, Vol. 9, Issue 6, June 1990, pp. 567-572.

[8] Hamdioui, S.; Van de Goor, A.J.; Rodgers, M.:"March SS: A Test for All Static Simple RAM Faults", IEEE International Workshop on Memory Technology, Design and Testing, Isle the Bendor, France, July 2002, pp. 95-100.

[9] Marinescu, M.: "Simple and Efficient Algorithms for Functional RAM Testing", IEEE International Test Conference, Philadelphia, PA, USA, November1982, pp. 236-239.

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

[11] Hamdioui, S.; Al-Ars, Z.; Van de Goor, A.J.: "Testing Static and Dynamic Faults in Random Access Memories", 20th IEEE VLSI Test Symposium, Monterey, CA, USA, May 2002, pp. 395-400

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

[13] Van de Goor, A.J.; Gayadadjiev, G.N.; Yarmolik, V.N.; Mikitjuk, V.G.: "March LA: A Test for Linked Memory Faults", European Design and Test Conference, Paris, France, March 1997, pp. 167.

[14] Van de Goor A.J.; Gayadadjiev, G.N.; Yarmolik, V.N.; Mikitjuk, V.G.: "March LR: A Test for Realistic Linked Faults", 16th IEEE VLSI Test Symposium, Princeton, NJ, USA, May 1996, pp. 272-280.

[15] Al-Harbi, S.M.; Gupta, S.K.: "Generating Complete and Optimal March Tests for Linked Faults in Memories", IEEE VLSI Test Symposium, Napa Valley, CA, USA, April 2003, pp. 254 -261.

[16] Hamdioui, S. ; Al-Ars, Z.; Van de Goor, A.J., Rodgers, M.: "March SL: a test for all static linked memory faults", 12th IEEE Asian Test Symposium, Xian, China, May, 2003. pp. 372 – 377.

[17] Hamdioui, S.; Al-Ars, Z.; Van de Goor, A.J.; Rodgers, M.: "Linked Faults in Random Access Memories Concept Fault Models Test Algorithms and Industrial Results", IEEE Transaction on CAD, Vol.: 23, Issue: 5, May 2004, pp. 737-757.

[18] Harutunyan, G.; Vardaninan, V.A.; Zorian, Y.: "Minimal March Test Algorithm For Detection Of Linked Static Faults In Random Access Memories", 24th IEEE VLSI Test Synposium, Berkeley, California, USA, April 2006, pp. 120-125.

[19] Benso, A.; Bosio, A.; Di Carlo, S.; Di Natale, G.; Prinetto, P.: "Automatic March tests generation for static and dynamic faults in SRAMs", IEEE European Test Symposium, Tallinn, Estonia, May 2005, pp. 122-127.

[20] Benso, P.; Di Carlo, S.; Di Natale, G.; Prinetto, P.: "Specification and design of a new memory fault simulator", 11th IEEE Asian Test Symposium, Guam, USA, November 2002. pp. 92-97.