A 22n March Test for Realistic Static Linked Faults in SRAMs

A 22n March Test for Realistic Static Linked Faults in SRAMs

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

Linked Faults are considered an interesting class of memory faults. Their capability of influencing the behavior of other faults causes the hiding of the fault effect and makes test algorithm design a very complex task. Although several March Tests have been developed for the wide memory faults spread, a few of them are able to detect linked faults. In the present paper March AB, a March Test targeting the set of realistic memory linked fault is presented. Comparison results show that the proposed March Test provides the same fault coverage of already published algorithms but, it reduces the test complexity and therefore the test time. Moreover, a complete taxonomy of linked faults will be presented.

1. Introduction

Memories are one of the most important components in digital systems, and semiconductor memories are nowadays one of the fastest growing technologies. Actually the major trend of *System-On-a-Chip* (SOC) allows to embed in a single chip all the components and functions that historically were placed on a hardware board. Within SOCs, embedded memories are the densest components, accounting for up to 90% of chips area [1]. 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.

The high density of their cells array makes memories extremely vulnerable to physical defects. Due to the complex nature of the internal behaviour of memory chips, the design of fault models and tests is non-trivial.

A linked fault is a memory fault composed of two or more simple faults. The behaviour of each simple fault can be influenced by the remaining ones and in some cases the fault can be masked. Classic March tests cannot detect linked faults due to the masking effect.

In the latest decade published researches mainly focused on the definition of new fault models [2] [3] [4] [5] showing the importance of developing new memory test algorithms. Nevertheless a few publications targeted the problem of linked faults.

March A, March B [6], March LA [7], and March LR [8] have an high fault coverage on a restricted set of linked memory faults. In [9], the authors present an automatically generated March algorithm of a complexity of 43*n*. This March test is still affected by the problem of detecting a limited number of memory faults, the same of [6], [7], and [8]. In [11] and [10] the authors present an accurate analysis of the linked fault concept, they also present a March test facing new fault models. The presented March SL has a complexity of 41*n*.

In this paper we present March AB, a March test targeting the same set of faults already covered by March SL, but reducing the test complexity of the best previous work by 54 % or by 19*n*.

To better identify the target faults, a taxonomy of realistic linked faults is presented, and each addressed fault is modeled resorting to the Fault Primitive formalism introduced in [12]. To analytically prove the efficiency of the proposed March Test, for each fault model the coverage conditions, i.e. the sequence of memory operations needed to sensitize and detect the fault effects, are defined. Moreover, 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 algorithms. The correctness of the proposed test has been also proved by fault simulation experiments performed by using an in-house developed memory fault simulator [13].

The paper is structured as follows: Section 2 introduces the fault model formalism, Section 3 introduces the concept of the linked fault and its relative taxonomy; Section 4 presents the new March Test and the complete list of the coverage conditions is detailed in Section 5. Section 6 validates the proposed algorithm Comparisons evaluations are reported in Section 7, while Section 8 summarizes the main contributions and outlines future research activities.

2. Fault Model

For test purposes, faults in memories are usually 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. A FFM involves one or more *Faulty Memory Cells* (FMC)

Proceedings of the Eleventh IEEE European Test Symposium (ETS'06)
0-7695-2566-0/06 $20.00 © 2006 IEEE

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. Each FFM can be described by a set of Fault Primitives (FPs) [12]. A Fault Primitive is identified by <S/F/R>, it represents the difference between an expected (good), and the observed (faulty) memory behavior; in which:

- *S = Sa ; [Sv]* is a sequence of *m* operations and/or conditions, respectively applied to *a*-cell (*Sa*)and *v*-cell (*Sv*), needed to sensitize the given fault. The *j*-th operation is represented as $op_j = iOd_j$ where $i \in \{0,1\}$ is the initial value stored in a memory cell; $O \in \{w,r\}$ is the type of operation performed on a cell; $d \in \{0,1\}$ in case of write operation represents the data to be written into memory *cell*. *Sv* is omitted when the FP correspond to a single cell memory fault, because it involves just one cell
- *F* is the faulty behavior, i.e., the value (state) stored in the victim cells after applying *S*
- *R* is the sequence of values read on the aggressor cell when applying *S*.

As an example FP = <0w1 ; 0/1/- > means that the operation 'w1' performed on the *a*-cell, when the initial state is 0 for both *a* and *v* cells, causes the *v*-cell to flip.

Several FPs classification rules can be adopted, based on the number of memory operations (*m*) needed to sensitize the FP (*static* when *m* = 1 or *dynamic* fault elsewhere); and based on the number of memory cells (#FMC) involved by the FP (*single cell* where #FMC = 1 or *n-cells* elsewhere fault) [12]. Hereinafter we deal with static faults (i.e., *m* = 1) that have been proved to be the most realistic fault models when linked [11].

3. Linked Fault: Concept & Taxonomy

In some cases it is possible that the effect of a FFM influences another functional fault. If these faults share the same aggressor and/or victim cells, the FFMs are called *Linked*, otherwise they are called simple or un-linked and each fault is independent from the others. To understand the concept of linked faults we can consider, as an example, the Disturb Coupling Faults [12] described by the following two FPs:

$$FP_1 = < 0w1 ; 0 / 1 / - >, FP_2 = < 0w1 ; 1 / 0 / - > \quad (1)$$

The most general case is represented in Figure 1, in which a *n* cells memory is affected by two FPs (FP1 and FP2) having different *a*-cells (a_1, a_2) and the same *v*-cell. The vertical arrow shows the address order of the memory (from the lowest memory address to the highest) in which *i*, *j* and *k* represent the address of a_1, a_2 and *v*, respectively. By first performing "0w1" (FP1) on cell *i*, the *v*-cell *k* flips from 0 to 1; than performing "0w1" (FP2) 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 FP2, since FP2 has a fault effect (F) opposite to FP1.

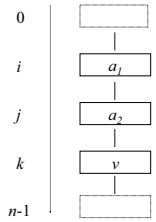

Figure 1. Example of Linked Fault

Looking at the example of Figure 1, we can derive a rigorous definition of a Linked Fault (LF):

Definition 1 : two FPs, FP1 = <S1/F1/R1> and FP2 = <S2/F2/R2>, are said to be *Linked,* and denoted by "FP1 → FP2", if both of the following conditions are satisfied:

- FP2 masks FP1, i.e., F2 = not (F1);
- The Sensitizing operation (S_2) of FP2 is applied after S1, on either the *a*-cell or *v*-cell of FP1.

To detect linked faults (LFs), one must detect in isolation at least one of the FPs that compose the fault (i.e., without allowing the other FP to mask the fault) [11].

In the sequel, we detail the taxonomy of the realistic LFs. The classification is based on the number of memory cells involved by the fault. We consider only realistic faults involving one (single cell LF), two and three cells. These faults have been proved to be the most realistic memory linked faults [11].

3.1. Realistic Single cell Linked Faults

The single cell Linked Faults involve a single memory location in which all the FPs are sequentially applied. The set of realistic single cell Linked faults, reported in Table 1, has been published and validated in [11]. Table 1 reports the whole set of single cell LFs, for each linked fault, the FP formalism with compact notation describe the fault. Compact notation resorts to *x*, *y* variable where $x, y \in \{0,1\}$, x = not (y).

Table 1 Single Cell LFs

Linked Fault	*FPs*	*S1*	*S2*
TF→WDF	<S1 / x / -> → <S2 / y / ->	xwy	xwx
WDF→WDF	<S1 / x / -> → <S2 / y / ->	ywy	xwx
DRDF→WDF	<S1 / x / y> → <S2 / y / ->	yry	xwx
TF→ RDF	<S1 / x / -> → <S2 / y / y>	xwy	xrx
WDF→ RDF	<S1 / x / -> → <S2 / y / y>	ywy	xrx
DRDF→ RDF	<S1 / x / -> → <S2 / y / y>	yry	xrx

Table 2 Realistic Two cells LF2aa: z, l, j, x, y ∈ {0,1}, x = not(y), l = not(j)

Linked Fault	*FPs*	*S1*	*S2*
CFds→CFds	<S1 ; x / y /-> → <S2 ; y / x /->	lwj , jwj, jrj	jwl , jwj , jrj
CFtr→CFds	<z ; S1 / x /-> → <S2 ; x / y /->	xwy	jwl
CFwd→CFds	<z ; S1 / y /-> → < S2 ; y / x /->	xwx	jwl
CFdr→CFds	<z ; S1 / y /x > → <S2 ; y / x /->	xrx	jwl
CFds→CFwd	<S1 ; x / y /- > → <z ; S2 / x /->	jwl , jwj , jrj	ywy
CFtr→CFwd	<z ; S1 / x /- > → <z ; S2 / y /->	xwy	xwx
CFwd→CFwd	<z ; S1 / x /- > → <z ; S2 / y /->	ywy	xwx
CFdr→CFwd	<z ; S1 / x /y > → <z ; S2 / y /->	yry	xwx
CFds→CFrd	<S1 ; x / y /- > → <z ; S2 / x / x>	jwl , jwj , jrj	yry
CFtr→CFrd	<z ; S1 / y /- > → <z ; S2 / x / x>	ywx	yry
CFwd→CFrd	<z ; S1 / y /- > → <z ; S2 / x / x>	xwx	yry
CFdr→CFrd	<z ; S1 / y /x > → <z ; S2 / x /x>	xrx	yry

Table 3 Realistic Two cells $LF2_{av}$: z, l, j, x, y ∈ {0,1}, x = not(y), l = not(j)

Linked Fault	*FPs*	*S1*	*S2*
CFds→WDF	<S1 ; x / y /-> → <S2 / x /->	jwl , jwj , jrj	ywy
CFtr→ WDF	<z ; S1 / y /-> → <S2 / x /->	ywx	ywy
CFwd→WDF	<z ; S1 / y /-> → < S2 / x /->	xwx	ywy
CFdr→WDF	<z ; S1 / y /x > → <S2 / x /->	xrx	ywy
CFds→RDF	<S1 ; x / y /- > → <S2 / x /x>	jwl , jwj , jrj	yry
CFtr→RDF	<z ; S1 / y /- > → <S2 / x /x>	ywx	yry
CFwd→RDF	<z ; S1 / y /- > → <S2 / x /x>	xwx	yry
CFdr→RDF	<z ; S1 / y /x > → <S2 / x /x>	xrx	yry

Table 4 Realistic Two cells $LF2_{va}$: z, l, j, x, y ∈ {0,1}, x = not(y), l = not(j)

Linked Fault	*FPs*	*S1*	*S2*
WDF→CFds	<S1 / x /->→<S2 ; x / y /->	ywy	jwl , jwj , jrj
TF→CFds	<S1 / x /->→<S2 ; x / y /->	xwy	jwl , jwj , jrj
DRDF→CFds	<S1 / x /y >→<S2 ; x / y /->	yry	jwl , jwj , jrj
WDF→CFwd	<S1 / x /->→<z ; S2 / y /->	ywy	xwx
TF→CFwd	<S1 / x /->→<z ; S2 / y /->	xwy	xwx
DRDF→CFwd	<S1 / x /y >→<z ; S2 / y /->	yry	xwx
WDF→CFrd	<S1 / x /->→<z ; S2 / y /y >	ywy	xrx
TF→CFrd	<S1 / x /->→<z ; S2 / y /y >	xwy	xrx
DRDF→CFrd	<S1 / x /y >→<z ; S2 / y /y >	yry	xrx

3.2. Realistic Two cells Linked Faults

Two cells Linked Faults involve two distinct memory cells: one aggressor cell, and one victim. Figure 2 shows the possible mutual positions of the two involved cells.

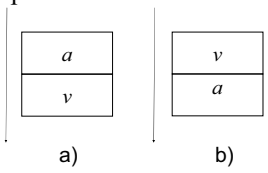

Figure 2. Two cells LFs; a) *a* < *v*, b) *v* < *a*

The set of realistic two cells Linked faults can be found in [11]. Two linked FPs "FP1→FP2" can be clustered in three different classes of realistic faults:

- $LF2_{aa}$: LFs that share both the a-cells and v-cells (Table 2);
- $LF2_{av}$: LFs where FP1 is a two cells FP and FP2 is a single cell FP (Table 3);
- $LF2_{va}$: LFs where FP1 is the single cell FP and FP2 is the two cells FP (Table 4).

From Table 2 to Table 4 the whole set of realistic two cell LFs is exploited, we formalize each FPs by using compact notation, where *z, j, l, x, y* ∈ {0,1}, *x* = not (*y*) and *l* = not(*j*).

3.3. 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). The realistic fault model [11] is shown in Figure 3, where the *v*-cell is between the *a*-cells. Realistic three cell space is exactly the same as two cell LFs (Table 2, Table 3 and Table 4)

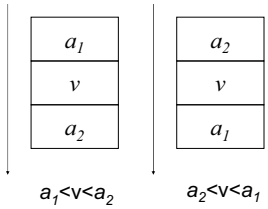

Figure 3. Three cells LFs.

4. March Test

A *March Test* is a test algorithm composed of a sequence of *March Elements* [14]. Each *March Element* (ME) 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 the ME. Hereinafter, we shall 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. Figure 4 shows the **March AB**, with a complexity of 22*n.* March AB is able to detect the whole set of realistic static linked faults (Section 3). Compared with March SL [10], the state-of-the-art algorithm to target the same set of faults with a complexity of 41*n*, March AB reaches the same fault coverage but reduces the test length of about 54% or by 19*n*.

We obtain the new March tests AB by using the automatic March test generation algorithm introduced in [15], moreover generation process also allows the definition of a set of *Fault Coverage Conditions* (FCC) needed to detect the target faults. Each FCC specifies the March Elements able to detect the target fault. In the sequel of the paper, we will introduce the coverage conditions for the set of linked faults listed in Section 3 and we will prove that March AB satisfies all those conditions.

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

Figure 4. March AB *O*(*n*) = 22*n*

5. Fault Coverage Conditions

A *Fault Coverage Conditions* (FCC) represents a MEs sequence, formalized with the March Test notation (Section 4). It can be automatically derived from the FP formalism by using March test generation algorithm introduced in [15], that simply builds the set of March tests targeting each FPs. Then we compare March AB whit the FCCs and we check the occurrence of the FCCs inside March AB in order to ensure the correctness of our algorithm. Next sections detail the FCCs and prove their coverage.

5.1. Single cell LF Detection

Single cell linked faults are sensitized and detected by performing operations on the FMC. As described in section 3.1, two main classes can be composed having the same FP2. The first one has FP2 equal to WDF and FP1 = {TF, WDF} (Table 1). In this class, FP1 is sensitized by S1 = {xwy, ywy, yry} and FP2 by S2= xwx. ME = $\Updownarrow(r_y,w_x,r_x,w_x,r_x)$ detects WDF in isolation by applying fourth operation (w_x) that sensitize the fault, then read (r_x) observes the fault effect. FP1 cannot be sensitized since the ME doesn't contain any operations belonging to S1.

Second class has FP2 equal to RDF and FP1 = {TF, WDF, DRDF} (Table 1). Here FP2 is sensitized by S2 = xrx and FP1 by S1 = {x*w*y, ywy, yry}, therefore ME = $\Updownarrow(r_x,w_y,r_y,w_y,r_y)$ detects RDF in isolation by the first operation (r_x), FP1 is sensitized after FP2 therefore masking cannot occur. The two FCCs covering the entire set of single cell LFs are:

FCC1 = $\Updownarrow(r_y,w_x,r_x,w_x,r_x)$, **FCC2** = $\Updownarrow(r_y,w_x,r_x,w_x,r_x)$

5.2. Two cells LF Detection

The two cells LFs detection is more complex than those for single cell, because the relations between *aggressor* and *victim* address constraint (i.e., $a < v$ and $v > a$) must be respected. Referring to of two cells LFs classification (Section 3.2), we rank LFs having the same FP2. In the first group of faults, where FP1 = FP2 = CFds (Table 2, first row), FP2 is sensitized by S2 = {jwl , jwj , jrj}. We investigate each operations belong to S2, and the relative LF.

The first instance S2 = jwl imply LF = <S1 ; x / y /-> → < jwl ; y / x /->, where S1 = {lwj , jwj, jrj} and j,l,x,y ={0,1}; y = not (x), j = not(l).

Fixing the values: j = y and l = x, S1 = {xwy, ywy, yry} and S2 = ywx, the following ME can detect FP2 in isolation when a > v

$$\Downarrow(r_y,w_x,r_x,w_x,r_x)$$

In this case the first accessed cell is the a-cell, only FP2 can be sensitized since the v-cell is in y state, therefore second operation (w_x) sensitizes the fault (FP2), other faults cannot be sensitized, so when v-cell is accessed, the first read (r_y) detects the fault. In the same way a < v requires $\Uparrow(r_y,w_x,r_x,w_x,r_x)$.

Fixing the opposite values: j = x and l = y, S1 = {ywx , xwx, xrx} and S2 = xwy, the following MEs can detect FP2 in isolation when a < v

$$\Downarrow(r_x,w_y,r_y,w_y,r_y)\ \Updownarrow(r_{y,\ldots})$$

v-cell is firstly accessed and it sets the *y* state (w_y) on the v.cell. Then a-cell is accessed and the second operation (w_y) sensitizes the fault in isolation (FP2), other faults cannot be sensitized, since the ME doesn't include the required operations. First read (r_y) on the following ME detect the fault effect. Similarly a > v requires $\Uparrow(r_x,w_y,r_y,w_y,r_y)\ \Updownarrow(r_{y,\ldots})$

The second instance S2 = jrj imply LF = <S1 ; x / y /-> → < jrj; y / x /->, where S1 = {lwj , jwj, jrj} and j,l,x,y ={0,1}; y = not (x), j = not(l).

Fixing the values: j = y and l = x, S1 = {xwy, ywy, yry} and S2 = yry, then the following MEs can detect FP2 in isolation when a > v

$$\Downarrow(r_y,w_x,r_x,w_x,r_x)$$

accessing a-cell, the first operation (r_y) sensitizes the fault (FP2) in isolation. When v-cell is accessed, the first read (r_y) detect the fault. In the same way a < v requires $\Uparrow(r_y,w_x,r_x,w_x,r_x)$

setting the opposite values: j = x and l = y, S1 = {ywx , xwx, xrx} and S2 = xrx, then the following MEs can detect FP2 in isolation when a < v

$$\Downarrow(r_x,w_y,r_y,w_y,r_y)\ \Updownarrow(r_{y,\dots})$$

v-cell is firstly accessed, it sets the y state (w_y) on the v-cell. Then a-cell is accessed and the first operation (r_x) sensitizes the fault in isolation (FP2). First read on the follows ME detects the fault effect. In the same way a > v requires $\Uparrow(r_x,w_y,r_y,w_y,r_y)\ \Updownarrow(r_{y,\dots})$

Last instance S2 = jwj imply LF = <S1 ; x / y /-> → < jwj ; y / x /->, where S1 = {lwj , jwj, jrj} and j,l,x,y ={0,1}; y = not (x), j = not(l).

Fixing the values: j = y and l = x, S1 = {xwy, ywy, yry} and S2 = ywy, the following MEs detect FP2 in isolation when a < v

$$\Downarrow(r_x,w_y,r_y,w_y,r_y)\ \Updownarrow(r_{y,\dots})$$

v-cell is firstly accessed and it sets the y state (w_y) on the v.cell. Then a-cell is accessed and the fourth operation (w_y) sensitizes the fault in isolation (FP2). First read on the follows ME detect the fault effect. When S1 = yry, the last operation (r_y) will mask the fault. In order to avoid this conditions, the second ME is refined as $\Downarrow(r_y,w_x,r_x,w_x,r_x)\ \Updownarrow(r_{x,\dots})$, where FP1 is sensitized by the first operation and observed by the third ME. In the same way a > v requires $\Uparrow(r_x,w_y,r_y,w_y,r_y)\ \Uparrow(\ r_y,w_x,r_x,w_x,r_x)\ \Updownarrow(r_{x,\dots})$. Fixing the opposite values: j = x and l = y, S1 = {ywx , xwx, xrx} and S2 = xwx, then the following ME detects FP2 in isolation when a > v

$$\Downarrow(r_y,w_x,r_x,w_x,r_x)$$

a-cell is firstly accessed; the fourth operation (w_x) sensitizes the fault in isolation (FP2). First read on the v-cell will detect the fault effect. When S1 = xrx, the last operation (r_y) will mask the fault. It requires the following ME $\Downarrow(r_x,w_y,r_y,w_y,r_y)$ in order to sensitize and detect in isolation FP1 In the same way a < v requires $\Uparrow(r_y,w_x,r_x,w_x,r_x)\ \Uparrow(r_x,w_y,r_y,w_y,r_y)$

Finally the full set of CFds linked to CFds is detected by

$$\textbf{FCC3}: \Downarrow(r_x,w_y,r_y,w_y,r_y)\Downarrow(r_y,w_x,r_x,w_x,r_x)$$
$$\Uparrow(r_x,w_y,r_y,w_y,r_y)\Uparrow(r_y,w_x,r_x,w_x,r_x)\ \Updownarrow(r_{x,\dots})$$

If FP1 is a CFtr, a CFwd, or a CFdr, it is easy to see that FCC3 is still able to detect them, since each CFds is detected in isolation. Similarly it is possible to verify that the detection conditions for the remaining LF2aa, LF2av (Table 3) and LF2va (Table 4), still remain FCC3 that also cover the remaining LF2s

5.3. Three cells LF Detection

Three cells LFs are composed of two cells FPs (see Section 3.3) sharing the same *v*-cells but having different *a*-cells (a_1 and a_2). [11] proves that the conditions detecting two cells LFs are enough to detect all the three cells LFs. Therefore, FCC3 ensure the detection of the entire three cells LFs space.

6. March AB Validation

In order to validate March AB we have to prove that it includes the FCCs introduced in Section 5. First of all it is immediately clear that FCC1 and FCC2 (Table 5) are included in FCC3. In other word FCC3 still cover single cell LFs. We can expand FCC3 exploiting the whole set of value assumed by *x* and *y*. Table 6 shows each ME obtained by a couple of x,y value. Looking the results we note that some March elements are redundant, in particular M1 = M6, M2 = M7, M3 = M8 and M4 = M9. Note that M10 is included in M4. After removing the redundancy we obtain five MEs shown in Table 7. It is now trivial task to prove that FCC3 correspond to March AB (Figure 4)

7. Comparing March Tests

We compared our test with March SL [10] since both target the same set of Linked faults. We also considered others March Tests (A, B, LR, LA and [9]) still addressing linked faults, but targeting a reduced set of fault models, in particular a subset of the FFMs presented in Section 3. Each March algorithm has been simulated using the memory fault simulator presented in [13]. Table 8 summarizes the simulation results in terms of fault percentage covered by each March Test and its complexity (O(*n*)). It targets single cell LFs, two cells LFs and three cells LFs. Comparison results show that the proposed March Test provides the same fault coverage of the best known one, but it reduces the test complexity, and therefore the test time of a significantly 54%. Furthermore, [16] proves that March AB covers

Table 5 expanded FCC3

#	March Elements	*x,y*
M1	$\Downarrow(r_0,w_1,r_1,w_1,r_1)$	$x = 0, y = 1$
M2	$\Downarrow(r_1,w_0,r_0,w_0,r_0)$	$x = 0, y = 1$
M3	$\Uparrow(r_0,w_1,r_1,w_1,r_1)$	$x = 0, y = 1$
M4	$\Uparrow(r_1,w_0,r_0,w_0,r_0)$	$x = 0, y = 1$
M5	$\Updownarrow(r_{0,...})$	$x = 0, y = 1$
M6	$\Downarrow(r_0,w_1,r_1,w_1,r_1)$	$x = 1, y = 0$
M7	$\Downarrow(r_1,w_0,r_0,w_0,r_0)$	$x = 1, y = 0$
M8	$\Uparrow(r_1,w_0,r_0,w_0,r_0)$	$x = 1, y = 0$
M9	$\Uparrow(r_0,w_1,r_1,w_1,r_1)$	$x = 1, y = 0$
M10	$\Updownarrow(r_{1,...})$	$x = 1, y = 0$

Table 6 reduced FCC3

#	March Elements
M1	$\Downarrow(r_0,w_1,r_1,w_1,r_1)$
M2	$\Downarrow(r_1,w_0,r_0,w_0,r_0)$
M3	$\Uparrow(r_0,w_1,r_1,w_1,r_1)$
M4	$\Uparrow(r_1,w_0,r_0,w_0,r_0)$
M5	$\Updownarrow(r_{0,...})$

Table 7 Simulation Results

MT	*O (n)*	*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%
SL	41*n*	100%	100%	100%	100%	100%
[9]	43*n*	83%	84%	83%	86%	84%

The whole set of static and dynamic unlinked faults, making possible resort to a single March test able to detect the bigger set of realistic memory fault, therefore March AB becomes a natural candidate for memory BIST architectures, building our test solution very attractive for industry

8. Conclusions

This paper proposed March AB, a new March Test targeting static linked memory faults. The detailed analysis of the March algorithm proves the detection capability. Moreover we validated March AB by fault simulation experiments, showing that our test provides the same coverage of the state-of-the-art test algorithm (March SL) but reducing test complexity of 54% and therefore the test time. On going activities are focused on the definition and validation of new complex fault models, such as dynamic linked faults and multi-port memory faults, and their test solutions.

9. References

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

[2] 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.

[3] 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.

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

[5] 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.

[6] D. S. Suk, S. M. Reddy, *"A March Test for Functional Faults in Semiconductor Random-Access Memory"* IEEE Transaction on Computer-Aided Design, Volume: 30, Issue: 12, 1981.

[7] A. J. van de Goor, G.N. Gayadadjiev, V.N. Yarmolik, V.G. Mikitjuk, *"March LA: A Test for Linked Memory Faults",* ED&TC 1997, Proc. European Design and Test Conference, 1997, pp. 167.

[8] A. J. van de Goor, G.N. Gayadadjiev, V.N. Yarmolik, V.G. Mikitjuk, *"March LR: A Test for Realistic Linked Faults"*, VTS 1996, 16th IEEE VLSI Test Symposium, 1996, pp. 272-280.

[9] S.M. Al-Harbi, S.K. Gupta*, "Generating Complete and Optimal March Tests for Linked Faults in Memories",* VTS 2003, 21th IEEEVLSI Test Symposium, 2003, pp. 254 -261.

[10] S. Hamdioui, Z. Al-Ars, A.J. van de Goor, M. Rodgers, "*March SL: a test for all static linked memory faults"*, ATS 2003, 12th IEEE Asian Test Symposium, 2003. pp. 372 – 377.

[11] S. Hamdioui, Z. Al-Ars, A. J. van de Goor, M. Rodgers, *"Linked Faults in Random Access Memories Concept Fault Models Test Algorithms and Industrial Results"*, IEEE Transaction on Computer-Aided Design, Volume: 23, Issue: 5, May 2004, pp. 737-757.

[12] 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.

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

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

[15] A. Benso, A. Bosio, S. Di Carlo, G. Di Natale, P. Prinetto, "*Automatic March Tests Generation for Static Linked Faults in SRAMs*", DATE 2006, IEEE Design Automation and Test in Europ, 2006.

[16] A. Benso, A. Bosio, S. Di Carlo, G. Di Natale, P. Prinetto, "*March AB, March AB1: New March Tests for Unlinked Dynamic Memory Faults*", ITC 2005, IEEE International Test Conference, 2005.