March SS: A Test for All Static Simple RAM Faults

Said Hamdioui[1,2] Ad J. van de Goor[2] Mike Rodgers[1]

[1]Intel Corporation, 2200 Mission College Boulevard, Santa Clara, CA 95052
[2]Delft University of Technology, Faculty of Information Technology and Systems
Computer Engineering Laboratory, Mekelweg 4, 2628 CD Delft, The Netherlands
E-mail: said@ce.et.tudelft.nl

Abstract

This paper presents all simple (i.e., not linked) static fault models that have been shown to exist for Random Access Memories (RAMs), and shows that none of the current industrial march tests has the capability to detect all these faults. It therefore introduces a new test (March SS), with a test length of 22n, that detects all realistic simple static faults in RAMs.

Keywords: *Memory testing, fault models, simple/linked faults, march test, fault coverage.*

1 Introduction

Semiconductor memories are an integral part of modern ULSI circuits. With each new generations of ULSIs, the memory share of the chip area increases and is expected to be 94% in 2014 [13]. Hence, memory testing will become a major cost factor in the production of the modern ULSIs. Precise fault modeling and efficient test design, in order to keep test cost and time within economically acceptable limits, is therefore essential. The quality of the used tests, in terms of their fault coverage and test length, is strongly dependent on the used fault models.

Many *Functional fault models (FFMs)* for memories have been introduced in the past; some well known FFMs, which date back to before 1980, are address decoder faults and stuck-at faults [17]. Of later date are the following FFMs: data retention fault, stuck open fault [4], read destructive fault, deceptive read destructive fault [1], and disturb coupling fault [15]. In 1999, experimental results by applying a large number of tests to a large number of chips [18, 12] indicated that many functional tests do detect faults in memories, which cannot be explained using the well know set of FFMs. This means that additional FFMs exist. This has led to the introduction of new FFMs, based on defect injection and circuit simulation [2, 3, 6]: write disturb fault, incorrect read fault, transition coupling fault, read destructive coupling fault, etc.

The fact that new FFMs (which have been shown to exist in real designs) have been introduced calls for the evaluation of the fault coverage of the current industrial march tests and their capability for detecting the modern new FFMs. This paper deals with that subject. It first gives all FFMs that have been shown to exist. In addition, it shows that none of the industrial march tests detect all realistic faults. It therefore establishes a new test which covers all FFMs.

This paper is organized as follows. Section 2 describes the concept of a *fault primitive (FP)*, and classifies the memory faults. Section 3 uses the FP concept to define *static, simple* FFMs. Section 4 introduces a new test, March SS, and compares it with the well known tests; while Section 5 ends with the conclusions.

2 Memory faults classification

This section gives first the concept of a fault primitive that will be used to define the set of the targeted FFMs in the paper. Second, a classification of memory faults will be given and the scope of the paper will be shown.

2.1 Fault primitives

By performing a number of memory operations and observing the behavior of any component functionally modeled in the memory, functional faults can be defined as the deviation of the observed behavior from the specified one under the performed operation(s). Therefore, the two basic ingredients of any fault model are: (a) A list of performed memory operations, and (b) A list of corresponding deviations in the observed behavior from the expected one. Any list of performed operations on the memory is called an *operation sequence*. An operation sequence that results in a difference between the observed and the expected memory behavior is called a *sensitizing operation sequence (S)*. The

1087-4852/02 $17.00 © 2002 **IEEE**

observed memory behavior that deviates from the expected one is called the *faulty behavior (F)*.

In order to specify a certain fault, one has to specify the S, together with the corresponding faulty behavior F and the read result (R) of S in case it is a read operation. The combination of S, F and R for a given memory failure is called a *Fault Primitive (FP)* [19], and is denoted as $< S/F/R >$. S describes the sensitizing operation sequence that sensitizes the fault, F describes the value or the behavior of the faulty cell (e.g., the cell flips from 0 to 1), while R describes the logic output level of a read operation (e.g., 0) in case S is a read operation.

The concept of FPs allows for establishing a complete framework of all memory faults, since for all allowed operation sequences in the memory, one can derive all possible types of faulty behavior. In addition, the concept of an FP makes it possible to give a precise definition of a *functional fault model (FFM)* as it has to be understood for memory devices [19]: *a functional fault model is a non-empty set of fault primitives*.

2.2 Classification

Figure 1 shows the different classifications of FPs. They can be classified based on:

1. the number of *simultaneous* operations required in the S, into *single-port* and *multi-port* faults.
2. the way the FPs manifest themselves, into *simple* and *linked* faults.
3. the number of *sequential* operations required in the S, into *static* and *dynamic* faults.

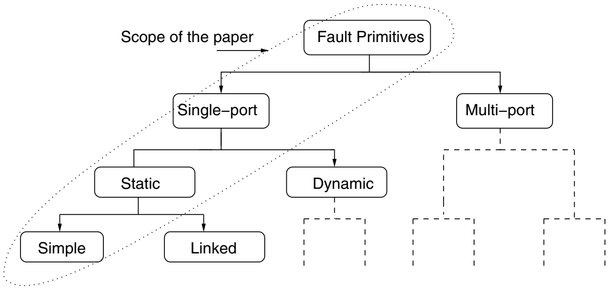

Figure 1. Fault primitive classification

It is important to note that the three ways of classifying FPs are independent since their definition is based on independent factors of the S; see Figure 1. As a result, a dynamic FP can be single-port or multi-port, simple or static. The same is true for linked faults; they can be static or dynamic, and each of them can be single-port or multi-port.

2.2.1 Single-port versus multi-port faults

Let $\#P$ be defined as the number of ports required *simultaneously* to apply a S. For example, if a single read operation applied to cell c_1 causes that cell to flip, then $\#P = 1$; if two *simultaneous* read operations applied to cell c_1 cause that cell to flip, then $\#P = 2$. Depending on $\#P$, FPs can be divided into *single-port* faults, and *multi-port* faults.

- *Single-port faults*: These are FPs that require *at the most* one port in order to sensitize a fault; that is $\#P \leq 1$. Note that single-port faults can be sensitized in single-port as well as in multi-port memories.
- *Multi-port faults:* These are FPs that can only sensitize a fault by performing two or more simultaneous operations via the different ports. Depending on $\#P$, the multi-port faults can be further divided into: (a) *Two-port faults* which can be only sensitized by performing two simultaneous operations via two different ports; (b) *Three-port faults* which can only be sensitized by performing three simultaneous operations via three different ports; etc. Testing multi-port faults is more complicated that testing single-port faults; they require specific patterns [7].

2.2.2 Static versus dynamic faults

Let $\#O$ be defined as the number of different operations performed *sequentially* in a S. For example, if a single read operation applied to a certain cell causes that cell to flip, then $\#O = 1$. Depending on $\#O$, FPs can be divided into *static* and *dynamic* faults:

- *Static faults*: These are FPs which sensitize a fault by performing *at the most* one operation; that is $\#O \leq 1$. For example, the state of the cell is always stuck at *one* ($\#O = 0$), a read operation to a certain cell causes that cell to flip ($\#O = 1$), etc.
- *Dynamic faults*: These are FPs that perform more than one operation *sequentially* in order to sensitize a fault; that is $\#O > 1$. Depending on $\#O$, a further classification can be made between *2-operation dynamic FPs* whereby $\#O = 2$, *3-operation dynamic FPs* whereby $\#O = 3$, etc. Experimental analysis of DRAMs, based on defect injection and SPICE simulation, shows that dynamic faulty behavior can take place in the absence of static faults [2, 3]. For example, two *successive* read operations cause the cell to flip; however, if only one read operation is performed, the cell will not flip. The current industrial march tests have been designed for static faults, and therefore may not be able to detect dynamic faults. All that indicates the importance of dynamic faults. Adequate fault models and tests for dynamic faults remain still to be established.

2.2.3 Simple versus linked faults

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

- *Simple faults*: These are faults which cannot be influenced by another fault. That means that the behavior of a simple fault cannot change the behavior of another one; therefore *masking* cannot occur.
- *Linked faults*: These are faults that do influence the behavior of each other. That means that the behavior of a certain fault can change the behavior of another one such that *masking* can occur [11, 17]. Note that linked faults consist of two or more simple faults. Although limited work has already been published about the subject of linked faults [15], the fault space for linked faults as well as the required tests remain still to be worked out.

In the remainder of this paper, we will focus on *single-port, static, simple faults*; see Figure 1. From here on, the term 'fault' refers to a single-port, static, simple fault'.

3 Fault models for RAMs

RAM faults can be divided into single-cell and multi-cell faults. Single-cell faults consist of FPs involving a single cell, while multi-cell faults consist of FPs involving more than one cell. For multi-cell FPs, we restrict our analysis to two-cell FPs (i.e., two-coupling FPs), because they are considered to be an important class for memory faults. Below single-cell FFMs and two-cell FFMs will be described.

3.1 Single-cell FFMs

Single-cell FFMs consist of FPs sensitized by performing at the most one operation to a single cell; i.e., the faulty cell. As mentioned in Section 2, a particular FP is denoted as $< S/F/R >$.

S describes the value/operation *sensitizing* the fault; $S \in \{0, 1, 0w0, 1w1, 0w1, 1w0, r0, r1\}$, where 0 (1) denotes a *zero* (*one*) value, $0w0$ ($1w1$) denotes a write 0 (1) operation to a cell which contains a 0 (1), $0w1$ ($1w0$) denotes an up (down) transition write operation, and $r0$ ($r1$) denotes a read 0 (1) operation.

F describes the value of the *faulty (i.e., victim)* cell (*v-cell*); $F \in \{0, 1, \uparrow, \downarrow\}$, where $\uparrow$ ($\downarrow$) denotes an up (down) transition due to a certain sensitizing *operation*.

R describes the logical value which appears at the output of the memory if the sensitizing operation applied to the v-cell is a *read* operation: $R \in \{0, 1, -\}$. A '$-$' in R means that the output data is not applicable; e.g., if $S = 1w0$, then no data will appear at the memory output, and therefore R is replaced by a '$-$'.

Now that all possible values of S, F, and R are known for single-cell faults, it is possible to list all detectable FPs using the notation $< S/F/R >$. It can easily be verified that there are 12 possible FPs [19]. These FPs are compiled into a set of six FFMs. They are listed in Table 1 together with their FPs:

1. *State Fault (SF)*: A cell is said to have a *state fault* if the logic value of the cell flips before it is accessed, even if no operation is performed on it[1]. This fault is special in the sense that no operation is needed to sensitize it and, therefore, it only depends on the initial stored value in the cell. The SF consists of two FPs: $< 0/1/- >$ and $< 1/0/- >$.
2. *Transition Fault (TF)*.
3. *Write Disturb Faults (WDF)*.
4. *Read Destructive Fault (RDF)* [1].
5. *Deceptive Read Destructive Fault (DRDF)* [1].
6. *Incorrect Read Fault (IRF)*.

Table 1. List of single-cell FFMs

#	FFM	Fault primitives
1	SF	$< 1/0/- >, < 0/1/- >$
2	TF	$< 0w1/0/- >, < 1w0/1/- >$
3	WDF	$< 0w0/\uparrow/- >, < 1w1/\downarrow/- >$
4	RDF	$< r0/\uparrow/1 >, < r1/\downarrow/0 >$
5	DRDF	$< r0/\uparrow/0 >, < r1/\downarrow/1 >$
6	IRF	$< r0/0/1 >, < r1/1/0 >$

3.2 Two-cell FFMs

Two-cell FFMs consist of FPs sensitized by performing at the most one operation while considering the effect two different cells have on each other. Such FPs can be presented as $< S/F/R > = < S_a; S_v/F/R >_{a,v}$, where S_a and S_v are the sensitizing operation sequences performed on the *aggressor (a-cell)* and the v-cell, respectively. The a-cell is the cell to which the sensitizing operation (or state) should be applied in order to sensitize the fault, while the v-cell is the cell where the fault appears. Note that in $< S_a; S_v/F/R >_{a,v}$, if S_a is an operation, then S_v should be a state; while if S_a is a state, then S_v can be a state or an operation. $S_a, S_v \in \{0, 1, 0w0, 1w1, 0w1, 1w0, r0, r1\}$. There are 36 possible FPs [19], which are compiled into seven FFMs. They are given in Table 2.

1. *State coupling fault (CFst)* [4]: Two cells are said to have a *state coupling fault* if the v-cell is forced into a given logic state only if the a-cell is in a given state, without performing any operation on the v-cell or on the a-cell. This fault is special in the sense that no operation is needed to sensitize it and, therefore, it only depends on the initial stored values in the cells. The CFst consists of four FPs: $< 0; 0/1/- >$, $< 0; 1/0/- >$, $< 1; 0/1/- >$, and $< 1; 1/0/- >$.

---

[1] It should be emphasized here that the state fault should be understood in the static sense. That is, the cell should flip in the short time period after initialization and before accessing the cell.

2. *Disturb coupling fault (CFds)* [15]: Two cells are said to have a *disturb coupling fault* if an operation (read, transition or non-transition write) performed on the a-cell causes the v-cell to flip. The CFds consists of 12 FPs.
3. *Transition coupling fault (CFtr).*
4. *Write Destructive coupling fault (CFwd).*
5. *Read Destructive coupling fault (CFrd).*
6. *Deceptive Read Destructive coupling fault (CFdrd).*
7. *Incorrect Read coupling fault (CFir).*

Table 2. List of two-cell FFMs; $x, y \in \{0, 1\}$

#	FFM	Fault primitives
1	CFst	$< 0; 0/1/- >, < 0; 1/0/- >,$ $< 1; 0/1/- >, < 1; 1/0/- >$
2	CFds	$< xwy; 0/\uparrow/- >, < xwy; 1/\downarrow/- >,$ $< rx; 0/\uparrow/- >, < rx; 1/\downarrow/- >$
3	CFtr	$< 0; 0w1/0/- >, < 1; 0w1/0/- >,$ $< 0; 1w0/1/- >, < 1; 1w0/1/- >$
4	CFwd	$< 0; 0w0/\uparrow/- >, < 1; 0w0/\uparrow/- >,$ $< 0; 1w1/\downarrow/- >, < 1; 1w1/\downarrow/- >$
5	CFrd	$< 0; r0/\uparrow/1 >, < 1; r0/\uparrow/1 >,$ $< 0; r1/\downarrow/0 >, < 1; r1/\downarrow/0 >$
6	CFdrd	$< 0; r0/\uparrow/0 >, < 1; r0/\uparrow/0 >,$ $< 0; r1/\downarrow/1 >, < 1; r1/\downarrow/1 >$
7	CFir	$< 0; r0/0/1 >, < 1; r0/0/1 >,$ $< 0; r1/1/0 >, < 1; r1/1/0 >$

It should be noted that all above single-cell and two-cell FFMs have been shown to exist [1, 2, 3, 4, 6, 8] in real designs. E.g., a SF can be caused in DRAMs (and SRAMs) by a short between the node (one of the nodes) of the cell and V_{cc} or V_{ss}; a TF can be caused in DRAMs and SRAMs by a broken pass transistor connection to the bit line; a WDF can be caused in DRAMs by a broken bit line, etc.

4 March tests

This section introduces a new march test, **March SS**, after which a comparison with current industrial march tests will be made. However, first the march notation will be given.

4.1 March notation

A complete march test is delimited by the '{...}' bracket pair, while a march element is delimited by the '(...)' bracket pair. March elements are separated by semicolons, and the operations within a march element are separated by commas. Note that all operations of a march element are performed at a certain address, before proceeding to the next address. The latter can be done in either one of two address orders: an increasing ($\Uparrow$) or a decreasing ($\Downarrow$) address order. When the address order is not relevant, the symbol $\Updownarrow$ is used.

4.2 March SS

March SS is shown in Figure 2. It has a test length of $22n$, and detects all single-cell and two-cell FFMs presented in Section 3. Minimization of the test length of the test was considered a high priority. However, M_5 can be extended (e.g., $\Updownarrow (r0, r0, w0, r0, w1)$) if a regular structure is required for BIST applications.

Let $M_{i,j}$ denote the j^{th} operation of march element M_i; e.g., $M_{1,3}$ denotes the third operation (i.e., w0) of M_1.

$$
\begin{array}{l}
\{ \ \underbrace{\Updownarrow (w0)}_{M_0} \ ; \\
\underbrace{\Uparrow (r0, r0, w0, r0, w1)}_{M_1} \ ; \ \underbrace{\Uparrow (r1, r1, w1, r1, w0)}_{M_2} \ ; \\
\underbrace{\Downarrow (r0, r0, w0, r0, w1)}_{M_3} \ ; \ \underbrace{\Downarrow (r1, r1, w1, r1, w0)}_{M_4} \ ; \\
\underbrace{\Updownarrow (r0)}_{M_5} \ \}
\end{array}
$$

Figure 2. March SS

4.2.1 Fault coverage of single-cell FFMs

March SS detects all single-cell FFMs:

- All SFs, RDFs and IRFs are detected since from each cell a 0 and a 1 is read.
- All TFs are detected because each cell is read after an up and a down transition write operation. The $< 0w1/0/- >$ is sensitized by $M_{1,5}$ (also by $M_{3,5}$) and detected by $M_{2,1}$ ($M_{4,1}$); while the $< 1w0/0/- >$ is sensitized by $M_{2,5}$ (also by $M_{4,5}$) and detected by $M_{3,1}$ (M_5).
- All WDFs are detected since each cell is read after a non-transition write operation; this is done by M_1 and M_2 (also by M_3 and M_4).
- All DRDFs are detected because two *successive* read operations are applied to each cell; the first read operation sensitizes the fault while the second detects it.

4.2.2 Fault coverage of two-cell FFMs

March SS detects all two-cell FFMs:

- The detection of CFst's requires that four states of any two cells can be generated and verified by a read operation [4]. The reader can easily verify by using a state diagram for March SS that all states of any two cells c_i and c_j (i.e., 00, 01, 11, 10) are generated and verified.
- All CFds's are detected; this include CFds's based on read operations, on transition write operations and on non-transition write operations. The first block of Table 3 shows by which march element (i.e., M_0 through M_5) of March SS, each FP belonging to each FFM is sensitized and detected. In the table, two cases have been distinguished: a) the v-cell has a higher address

than the a-cell (i.e., $v > a$), and b) the v-cell has a lower address than the a-cell ($v < a$). In addition, in each entry the notation Sensitization/Detection is used. E.g., the $< r0; 0/\uparrow/- >$ is sensitized and detected by $\mathrm{M}_{1,1}$ when $v > a$; while $< r0; 1/\downarrow/- >$ is sensitized by $\mathrm{M}_{3,1}$ and detected by $\mathrm{M}_{4,1}$ when $v > a$.

- All CFwd's are detected. The detection of CFwd's requires that each pair of cells undergoes the four states (00, 01, 10, 11), the application of a non-transition operation and thereafter a read operation. The second block of Table 3 shows by which march element each FP of CFwd is sensitized and detected.
- All CFdrd's, CFrd's, CFir's are detected. The detection of CFrd's and CFir's require that each pair of cells undergoes the four states (00, 01, 10, 11), and a read operation has to be performed to each of the two cell; while the detection of CFdrd's requires, in addition, the application of another read operation. Therefore, any test detecting CFdrd also detects CFrd and CFir. The third block of Table 3 shows by which march element each FP of CFdrd is sensitized and detected.
- All CFtr's are detected. The detection of CFtr's requires that each pair of cells undergoes the four states (00, 01, 10, 11), the application of a transition write operation to sensitize the fault, and thereafter a read operation to detect it. The fourth block of Table 3 shows by which march element each FP of CFtr is sensitized and detected.

Table 3. Fault coverage of March SS

FFM	FP	$v > a$	$v < a$
CFds	$< r0; 0/\uparrow/- >$	$\mathrm{M}_{1,1}/\mathrm{M}_{1,1}$	$\mathrm{M}_{3,1}/\mathrm{M}_{3,1}$
	$< r0; 1/\downarrow/- >$	$\mathrm{M}_{3,1}/\mathrm{M}_{4,1}$	$\mathrm{M}_{1,1}/\mathrm{M}_{2,1}$
	$< r1; 0/\uparrow/- >$	$\mathrm{M}_{4,1}/\mathrm{M}_{5}$	$\mathrm{M}_{2,1}/\mathrm{M}_{3,1}$
	$< r1; 1/\downarrow/- >$	$\mathrm{M}_{2,1}/\mathrm{M}_{2,1}$	$\mathrm{M}_{4,1}/\mathrm{M}_{4,1}$
	$< 0w1;0/\uparrow/- >$	$\mathrm{M}_{1,5}/\mathrm{M}_{1,1}$	$\mathrm{M}_{3,5}/\mathrm{M}_{3,1}$
	$< 0w1;1/\downarrow/- >$	$\mathrm{M}_{3,5}/\mathrm{M}_{4,1}$	$\mathrm{M}_{1,5}/\mathrm{M}_{2,1}$
	$< 1w0;0/\uparrow/- >$	$\mathrm{M}_{4,5}/\mathrm{M}_{5}$	$\mathrm{M}_{2,5}/\mathrm{M}_{3,1}$
	$< 1w0;1/\downarrow/- >$	$\mathrm{M}_{2,5}/\mathrm{M}_{2,1}$	$\mathrm{M}_{4,5}/\mathrm{M}_{4,1}$
	$< 0w0;0/\uparrow/- >$	$\mathrm{M}_{1,3}/\mathrm{M}_{1,1}$	$\mathrm{M}_{3,3}/\mathrm{M}_{3,1}$
	$< 0w0;1/\downarrow/- >$	$\mathrm{M}_{3,3}/\mathrm{M}_{4,1}$	$\mathrm{M}_{1,3}/\mathrm{M}_{2,1}$
	$< 1w1;0/\uparrow/- >$	$\mathrm{M}_{4,3}/\mathrm{M}_{5}$	$\mathrm{M}_{2,3}/\mathrm{M}_{3,1}$
	$< 1w1;1/\downarrow/- >$	$\mathrm{M}_{2,3}/\mathrm{M}_{2,1}$	$\mathrm{M}_{4,3}/\mathrm{M}_{4,1}$
CFwd	$< 0; 0w0/\uparrow/- >$	$\mathrm{M}_{3,3}/\mathrm{M}_{3,4}$	$\mathrm{M}_{1,3}/\mathrm{M}_{1,4}$
	$< 1; 0w0/\uparrow/- >$	$\mathrm{M}_{1,3}/\mathrm{M}_{1,4}$	$\mathrm{M}_{3,3}/\mathrm{M}_{3,4}$
	$< 0; 1w1/\downarrow/- >$	$\mathrm{M}_{2,3}/\mathrm{M}_{2,4}$	$\mathrm{M}_{4,3}/\mathrm{M}_{4,4}$
	$< 1; 1w1/\downarrow/- >$	$\mathrm{M}_{4,3}/\mathrm{M}_{4,4}$	$\mathrm{M}_{2,3}/\mathrm{M}_{2,4}$
CFdrd	$< 0; r0/\uparrow/0 >$	$\mathrm{M}_{3,1}/\mathrm{M}_{3,2}$	$\mathrm{M}_{1,1}/\mathrm{M}_{1,2}$
	$< 1; r0/\uparrow/0 >$	$\mathrm{M}_{1,1}/\mathrm{M}_{1,2}$	$\mathrm{M}_{3,1}/\mathrm{M}_{3,2}$
	$< 0; r1/\downarrow/1 >$	$\mathrm{M}_{2,1}/\mathrm{M}_{2,2}$	$\mathrm{M}_{4,1}/\mathrm{M}_{4,2}$
	$< 1; r1/\downarrow/1 >$	$\mathrm{M}_{4,1}/\mathrm{M}_{4,2}$	$\mathrm{M}_{2,1}/\mathrm{M}_{2,2}$
CFtr	$< 0; 0w0/\uparrow/- >$	$\mathrm{M}_{3,5}/\mathrm{M}_{4,1}$	$\mathrm{M}_{1,5}/\mathrm{M}_{2,1}$
	$< 1; 0w0/\uparrow/- >$	$\mathrm{M}_{1,5}/\mathrm{M}_{2,1}$	$\mathrm{M}_{3,5}/\mathrm{M}_{4,1}$
	$< 0; 1w1/\downarrow/- >$	$\mathrm{M}_{2,5}/\mathrm{M}_{3,1}$	$\mathrm{M}_{4,5}/\mathrm{M}_{5}$
	$< 1; 1w1/\downarrow/- >$	$\mathrm{M}_{4,5}/\mathrm{M}_{5}$	$\mathrm{M}_{2,5}/\mathrm{M}_{3,1}$

4.3 Comparison with other tests

Many memory test algorithms were developed to cover FFMs most of which had a theoretical origin. The traditional ad-hoc tests have been used in the past to screen the faulty devices. Walking 1/0, GALPAT, Butterfly, Zero-one test, and Checkerboard are widely known tests. However, the time complexity of the first two tests is completely unacceptable currently; while the fault coverage of the last three tests is not acceptable industrially.

March tests have been introduced to detect TFs, *Inversion Coupling Faults (CFin)*, *Idempotent Coupling Faults CFid (CFid)*, as well as SFs. A CFin is defined as: an up (or down) transition write operation in the a-cell causes an inversion in the v-cell; i.e., the v-cell flips to 0 if its content was 1, and flips to 1 if its content was 0. A CFid is defined as: an up (or down) transition write operation in the a-cell forces a certain fixed value, 0 or 1, in the v-cell. Table 4 summarizes the fault coverage of the following march tests:

- MATS+[10]: $\{\Updownarrow(w0); \Uparrow(r0, w1); \Downarrow(r1, w0)\}$
- March C- [9, 17]: $\{\Updownarrow(w0); \Uparrow(r0, w1); \Uparrow(r1, w0); \Downarrow(r0, w1); \Downarrow(r1, w0); \Updownarrow(r0)\}$
- March B [14]: $\{\Updownarrow(w0); \Uparrow(r0, w1, r1, w0, r0, w1); \Uparrow(r1, w0, w1); \Downarrow(r1, w0, w1, w0); \Downarrow(r0, w1, w0)\}$
- PMOVI [5]: $\{\Downarrow(w0); \Uparrow(r0, w1, r1); \Uparrow(r1, w0, r0); \Downarrow(r0, w1, r1); \Downarrow(r1, w0, r0)\}$
- March U [16]: $\{\Updownarrow(w0); \Uparrow(r0, w1, r1, w0); \Uparrow(r0, w1); \Downarrow(r1, w0, r0, w1); \Downarrow(r1, w0)\}$
- March LR [15]: $\{\Updownarrow(w0); \Downarrow(r0, w1); \Uparrow(r1, w0, r0, w1); \Uparrow(r1, w0); \Uparrow(r0, w1, r1, w0); \Uparrow(r0)\}$
- March SR [6]: $\{\Downarrow(w0); \Uparrow(r0, w1, r1, w0); \Uparrow(r0, r0); \Uparrow(w1); \Downarrow(r1, w0, r0, w1); \Downarrow(r1, r1)\}$

In the table, e.g., "a/b" denotes that the test detects 'a' of the 'b' FPs of the correspondent FFM. E.g., March C- detects both FPs of TF, while MATS+ detects just one of them. For two-cell FFMs, each FP is divided into two sub-FPs: the a-cell has a higher address than the v-cell, and (b) the a-cell has a lower address than the v-cell. For example CFst consists of 4 FPs (see table 2); considering the position of the a-cell against the v-cell leads to 8 sub-FPs. E.g., MATS+ detect 4 of 8 CFst sub-FPs. The CFds is divided into three types: CFds$_{rx}$ whereby the fault is caused by a read operation; CFds$_{xw\overline{x}}$ whereby the fault is caused by a transition write operation; and CFds$_{xwx}$ whereby the fault is caused by a non-transition write operation ($x \in \{0, 1\}$). The FFMs CFin and CFid are not included in the table. The CFin, which has a theoretical origin, has never been shown to exist in real designs; while the CFid describes the same faults as CFds$_{xw\overline{x}}$.

The Modified Algorithm Test Sequence (MATS+) has been developed to detect SF, and has a test length of $5n$, whereby n is the memory size. March C- was introduced to

Table 4. Fault coverage for different march tests

FFM	March Tests							
	MATS+ ($5n$)	March C- ($10n$)	March B ($17n$)	PMOVI ($13n$)	March U ($13n$)	March LR ($14n$)	March SR ($14n$)	March SS ($22n$)
SF	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
TF	1/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
WDF	0/2	0/2	0/2	0/2	0/2	0/2	0/2	2/2
RDF	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
DRDF	0/2	0/2	0/2	2/2	0/2	0/2	2/2	2/2
IRF	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
CFst	4/8	8/8	6/8	8/8	8/8	8/8	8/8	8/8
CFds$_{rx}$	3/8	8/8	7/8	8/8	8/8	8/8	8/8	8/8
CFds$_{xw\overline{x}}$	3/8	8/8	8/8	7/8	8/8	8/8	8/8	8/8
CFds$_{xwx}$	0/8	0/8	0/8	0/8	0/8	0/8	0/8	8/8
CFtr	2/8	8/8	4/8	8/8	8/8	8/8	8/8	8/8
CFwd	0/8	0/8	0/8	0/8	0/8	0/8	0/8	8/8
CFrd	4/8	8/8	4/8	8/8	8/8	8/8	8/8	8/8
CFdrd	0/8	0/8	0/8	0/8	0/8	0/8	6/8	8/8
CFir	4/8	8/8	4/8	8/8	8/8	8/8	8/8	8/8

detect SF, TF, as well as CFin and CFid. The test also detects the CFst and RDF. In addition, it can be shown that the test also detects the modern new FFMs of Section 3: IRF, CFds (which assume disturbs by transition write operations as well as by *read* operations), CFrd, and CFir. However, March C- cannot detect WDF, DRDF, CFds$_{xwx}$, CFwd and CFdrd. March B, which is an extension of March A [14], was designed to detect *linked* CFin and CFid. March B has a poor fault coverage for simple faults. PMOVI detects all single-cell faults except WDF; in addition it does not detect CFds$_{xw\overline{x}}$, CFds$_{xwx}$, CFwd and CFdrd. March U has a similar capability as March C-. March LR has been designed for linked faults; it has a similar fault coverage as March U. March SR has the best relative fault coverage.

It is clear from the table that none of the existing tests has the capability to detect the FFMs: WDF, CFds$_{xwx}$, CFwd and CFdrd. This proves the need of March SS.

5 Conclusions

In this paper a classification of memory faults has been made, and the complete set of simple static faults has been presented. These faults have been shown to exist in real designs. In addition, it has been shown that none of the existing tests (like March C-, PMOVI, etc) can detect the whole realistic set of simple static faults. Therefore, March SS with a test length of $22n$ has been proposed. The test detects all simple static faults.

References

[1] R.D. Adams and E.S. Cooley, "Analysis of a Deceptive Read Destructive Memory Fault Model and Recommended Testing", *In Proc. IEEE North Atlantic Test Workshop*, 1996.

[2] Z. Al-Ars, Ad J. van de Goor, "Impact of Memory Cell Array Bridges on the Faulty Behavior in Embedded DRAMs", *In Proc. of ATS*, pp. 282-289 , 2000.

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

[4] R. Dekker, *et al.,* "A Realistic Fault Model and Test Algorithms for Static Random Access Memories", *IEEE Trans. on Computer Aided Design,* **C9**(6), pp. 567-572, 1990.

[5] J.H. De Jonge and A.J. Smeulders, "Moving Inversions Test Pattern is Thorough, Yet Speedy", *In Comp. Design*, pp. 169-173, 1976.

[6] S. Hamdioui, A.J. van de Goor, "Experimental Analysis of Spot Defects in SRAMs: Realistic Fault Models and Tests", *In Proc. of Ninth Asian Test Symposium,* pp. 131-138, 2000.

[7] S. Hamdioui, A.J. van de Goor, "Thorough Testing of Any Multi-Port Memory with Linear Tests", *In IEEE Trans. of CAD*, V. 21, N. 2, pp. 217-231, Feb. 2002.

[8] V.K. Kim and T. Chen, "On Comparing Functional Fault Coverage and Defect Coverage for Memory Testing", *IEEE Trans. on CAD*, V. 18, N. 11, pp. 1676-1683, 1999.

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

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

[11] C.A. Papachristou and N.B. Saghal, "An Improved Method for Detecting Functional Faults in Random Access Memories", *IEEE Trans. on Computers,* C-34, N. 3, pp. 110-116, 1985.

[12] I. Schanstra and A.J. van de Goor, "Industrial evaluation of Stress Combinations for March Tests Applied to SRAMs", *In Proc. IEEE Int. Test Conference*, pp. 983-992, 1999.

[13] Semiconductor Industry Association, *The National Roadmap for Semiconductors*, 2000.

[14] D.S. Suk and S.M. Reddy, "A March Test for Functional Faults in Semiconductors Random-Access Memories", *IEEE Trans. on Computers*, C-30(12), pp. 982-985, 1981.

[15] A.J. van de Goor and G. Gaydadjiev, "March LR: A Memory Test for Realistic Linked Faults", *In Proc. IEEE VLSI Test Symposium,* pp. 272-280, 1996.

[16] A.J. van de Goor and G.N. Gayadadjiev, "March U: A Test for All Unlinked Memory Faults, *IEE Proc. of Circuits Devices and Systems*, V. 144, No. 3, pp. 155-160, 1997.

[17] A.J. van de Goor, "Testing Semiconductor Memories, Theory and Practice", *ComTex Publishing, Gouda, The Netherlands*, 1998. Web: http://cardit.et.tudelft.nl/˜ vdgoor

[18] A.J. van de Goor and J. de Neef, "Industrial Evaluation of DRAMs Tests", *In Proc. of Design Automation and Test in Europe*, pp. 623-630, March 1999.

[19] A.J. van de Goor and Z. Al-Ars, "Functional Fault Models: A Formal Notation and Taxonomy", *In Proc. of IEEE VLSI Test Symposium*, pp. 281-289, 2000.