Dynamic Faults in Random-Access-Memories: Concept, Fault Models and Tests

SAID HAMDIOUI

Intel Corporation, 2200 Mission College Boulevard, Santa Clara, CA 95052, USA;
Section of Computer Engineering, Faculty of Information Technology and Systems,
Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
said@ce.et.tudelft.nl

ZAID AL-ARS AND AD J. VAN DE GOOR

Section of Computer Engineering, Faculty of Information Technology and Systems,
Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands

MIKE RODGERS

Intel Corporation, 2200 Mission College Boulevard, Santa Clara, CA 95052, USA

Received May 23, 2002; Revised October 11, 2002

Editor: A. Ivanov

Abstract. The ever increasing trend to reduce DPM levels of memories requires tests with very high fault coverage and low cost. This paper describes an important fault class, called dynamic faults, that cannot be ignored anymore. The dynamic fault behavior can take place in the absence of the static fault behavior, for which the conventional memory tests have been constructed. The concept of dynamic fault will be established and validated for both dynamic and static Random-Access-Memories. A systematic way to develop fault models for dynamic faults will be introduced. Further, it will be shown that conventional memory tests do not necessarily detect its dynamic faulty behavior, which has been shown to exist in real designs. The paper therefore also presents new memory tests to target the dynamic fault class.

1. Introduction

The cost of testing memories increases rapidly with every new generation of memory chips [15]. 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 [23]. Of later date are the following FFMs: data retention fault, stuck open fault [8], read destructive fault, deceptive read destructive fault [2], and disturb coupling fault [26]. In 1999, experimental results by applying a large number of tests to a large number of chips [21, 25] 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 [3, 4, 9]: write disturb fault, incorrect read fault, transition coupling fault, read destructive coupling fault, etc. Researchers have also introduced tests to target memory faults, with different degree of success [1, 2, 6–9, 16–19, 22, 23, 27, 29]. However, most of the work published on memory testing focuses on faults sensitized by performing *at most one operation*; e.g., a write operation sensitizes a fault. These FFMs are called *static faults*.

This paper deals with a new class of memory faults called *dynamic faults*. Dynamic faults require the application of *more than one operation sequentially* in order to be sensitized. For example, a write operation followed *immediately* by a read operation causes the cell to flip; however, if only a write or a read operation is performed, the cell will not flip. The paper validates the existence of such a fault class for dynamic as well as for static Random-Access-Memories (RAMs). In addition, it introduces a systematic way to develop dynamic fault models. The paper shows the shortcoming in the fault coverage of the traditional tests, which were originally designed for static faults, and therefore introduces a new simple and efficient test for dynamic faults.

This paper is organized as follows. Section 2 introduces the concept of *fault primitives* that will be used to classify memory faults and define the dynamic fault space in Section 3. Section 4 describes the importance of dynamic faults and validates their existence of both static and dynamic RAMs. Section 5 discusses the testing of dynamic faults; it first shows that the tests designed for static faults do not necessarily detect dynamic faults; and therefore it introduces a new test with 100% fault coverage of the targeted dynamic faults. Section 6 ends with the conclusions.

2. Memory Fault Classification

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

2.1. Fault Primitive Concept

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). There-

fore, 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 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 S is a read operation. The combination of S, F and R for a given memory failure is called a *Fault Primitive (FP)* [24], and is denoted as $\langle S/F/R \rangle$. S describes the sensitizing operation sequence that sensitizes the fault (e.g., a read '0' operation (i.e., $r0$)), F describes the value or the behavior of the faulty cell (e.g., the cell flips from 0 to 1 (i.e., $\uparrow$)), while R describes the logic output level of a read operation (e.g., a wrong value 1) in case S is a read operation; this can be written as '$\langle r0/ \uparrow /1 \rangle$'.

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 [24]: *a functional fault model is a non-empty set of fault primitives*.

2.2. Classification

Fig. 1 shows a number of different ways to classify the FPs. They can be classified based on:

1. The number of *simultaneous* operations required in the S, into *single-port* and *multi-port* faults.

Fig. 1.   Fault primitive classification.
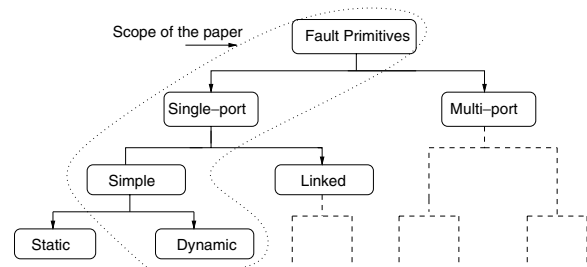
2. The way the FPs manifest themselves, into *simple* and *linked* faults.
3. The number of *sequential* operations required in the *S*, into *static* and *dynamic* faults.

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 Fig. 1. As a result, a dynamic FP can be single-port or multi-port, simple or linked. 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 via two different ports 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. This paper deals with single-port faults.
- *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 only be 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 than testing single-port faults; they require specific test patterns [10, 11].

2.2.2. 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. This paper deals with simple faults.
- *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 [20, 23]. Note that linked faults consist of two or more simple faults. In order to get more insight into linked faults, the following example will be given. Assume that the application of a sensitizing operation to a cell c_1 will cause a fault in a cell c_v (e.g., the cell flips); and that the application of an operation to a cell c_2 will cause a fault in the same cell c_v, but with a fault effect opposite to that caused by cell c_1. If an operation is first applied to cell c_1, and thereafter to cell c_2, then the net result is that the fault effect of cell c_1 is masked by the fault effect of cell c_2; i.e., no fault effect is then visible in cell c_v. Although limited work has already been published about the subject of linked faults [26, 28], the fault space for linked faults as well as the required tests remain still to be worked out.

2.2.3. 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, as will be described in Section 4, shows that dynamic faulty behavior can take place in the absence of static faults. 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; that is the subject of this paper.

In the remainder of this paper, we will focus on *single-port, simple, dynamic faults*; see Fig. 1. From

here on, the term 'fault' or 'dynamic fault' refers to a single-port, simple, dynamic fault'.

3. Dynamic Fault Space

Dynamic faults can be divided into FPs describing single-cell dynamic faults (involving a single-cell), and FPs describing multi-cell dynamic faults (involving more than one cell). For multi-cell FPs, we restrict our analysis to two-cell FPs, because they are considered to be an important class for memory faults [2–4, 9, 16]. Below single-cell dynamic faults and two-cell dynamic faults will be described.

3.1. Single-Cell Dynamic Faults

Single-cell dynamic faults consist of FPs sensitized by applying more than one operation to a single cell *sequentially*. We will restrict our analysis to 2-operation dynamic faults, since on the one hand they have been shown to exist [4, 13], and on the other hand the probability of dynamic faults decreases as the number of operations increases [5]. As mentioned in Section 2, a particular FP is denoted as $\langle S/F/R \rangle$.

S describes the *sensitizing operation sequence*, which sensitizes a fault *F* in the cell. Since two operations are considered, there are 18 possible *S*s. They are given below; $x, y, z \in \{0, 1\}$ and '*r*' denotes a read operation and '*w*' denotes a write operation.

- '$xwywz$'; e.g., '0w1r1' denotes a write 1 operation applied to a cell whose initial state is 0; the write is followed immediately with a read 1 operation.
- '$xrxrx$'; e.g., '0r0r0' denotes two successive read 0 operations applied to a cell whose initial state is 0.
- '$xrxwy$'; e.g., '0r0w1' denotes a read 0 followed immediately with write 1 applied to a cell whose initial state is 0.
- '$xwyry$'; e.g., '1w1r1 denotes a write 1 followed immediately with read 1 applied to a cell whose initial state is 1.

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* and 0 (1) denotes that the cell remains in its state 0 (1).

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: $R \in \{0, 1, -\}$.

Table 1. List of single-cell dynamic FFMs.

FFM	Fault primitives
dRDF	$\langle 0w0r0/\uparrow/1 \rangle$, $\langle 0w1r1/\downarrow/0 \rangle$,
	$\langle 1w0r0/\uparrow/1 \rangle$, $\langle 1w1r1/\downarrow/0 \rangle$
dDRDF	$\langle 0w0r0/\uparrow/0 \rangle$, $\langle 0w1r1/\downarrow/1 \rangle$,
	$\langle 1w0r0/\uparrow/0 \rangle$, $\langle 1w1r1/\downarrow/1 \rangle$
dIRF	$\langle 0w0r0/0/1 \rangle$, $\langle 0w1r1/1/0 \rangle$,
	$\langle 1w0r0/0/1 \rangle$, $\langle 1w1r1/1/0 \rangle$

A '$-$' in *R* means that the output data is not applicable; e.g., if $S = 0w0w1$, then no data will appear at the memory output, and for that reason *R* is replaced by a '$-$'.

In the following, only '$S = xwyry$' will be considered since, for instance, it has been shown to cause dynamic faults (see Section 4). Given that $S = xwyry$, $F \in \{0, 1, \uparrow, \downarrow\}$, and $R \in \{0, 1, -\}$, it can be verified that there are 12 possible FPs $\langle S/F/R \rangle$. These FPs are compiled into a set of three FFMs; they are listed in Table 1 together with their FPs:

1. *Dynamic Read Destructive Fault (dRDF)*: A write followed immediately by a read operation performed on a cell changes the data in the cell, and returns an *incorrect* value on the output. The dRDF consists of four FPs; e.g., $\langle 0w1r1/\downarrow/0 \rangle$: applying a '*r1*' operation immediately after '*w1*' operation to a cell whose initial content was 0, will cause the cell to flip to 0 and the read operation will return a wrong 0 value instead of the expected 1. The write operation involved in dRDF can be a transition write as well as a non-transition write operation.

2. *Dynamic Deceptive Read Destructive Fault (dDRDF)*: A write followed immediately by a read operation performed on a cell changes the data in the cell, and returns a *correct* value on the output. The dDRDF consists of four FPs. Here, the write can be a transition write as well as a non-transition write operation.

3. *Dynamic Incorrect Read Fault (dIRF)*: A read operation performed *immediately* after a write operation on a cell returns an *incorrect* value on the output, while the cell remains in its correct state. The dIRF consists of four FPs.

3.2. Two-Cell Dynamic Faults

Two-cell dynamic faults consist of FPs sensitized by applying more than one operation *sequentially* to two

cells: the *aggressor (a-cell)* and the v-cell. 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. In a similar way as it has been done for single cell faults, we will restrict ourself to two-operation dynamic faults. Then depending on how many operations are applied to the a-cell and to the v-cell, and on the order in which they are applied, four types of S can be distinguished:

1. S_{aa}: The two sequential operations are applied to the a-cell.
2. S_{vv}: The two sequential operations are applied to the v-cell.
3. S_{av}: The first operation is applied to the a-cell, followed immediately with a second one to the v-cell.
4. S_{va}: The first operation is applied to the v-cell, followed immediately with a second one to the a-cell.

Since two operations are considered, there are $18 \times 4 = 72$ total possible Ss; each S can take on one of 18 possible operation sequences: $xwywz$, $xrxrx$, $xrxwy$, or $xwyry$, where x, y, $z \in \{0, 1\}$. It is clear that despite the restriction to 2-operation dynamic faults, the number of FPs is still high. This calls for setting further restrictions on the sequences considered. For the simplicity, only S_{aa} and S_{vv} will be considered from now on.[1] Note that S_{aa} and S_{vv} both require the access of a single cell at a time (the v-cell, respectively the a-cell). Further, and as we did in the case of single-cell faults, the $S = xwyry$ will be considered; this sequence, for instance, has been verified to cause dynamic faults (see Section 4).

3.2.1. Faults Caused by S_{aa}.
The FPs for two-cell dynamic faults, where the two operations are applied to the a-cell, are represented as:

$$\langle S_{aa}/F/R \rangle = \langle S_a; S_v/F/R \rangle_{a,v}$$

S_a describes $xwyry$, while S_v describes the *state* of v-cell. Note that S_v does not specify an operation applied to the v-cell, but the *state* of the v-cell, since S_{aa} is considered. Now it can be verified easily that there are 8 possible FPs:

- 4 FPs denoted as $\langle xwyry; 0/ \uparrow /- \rangle$, and
- 4 FPs denoted as $\langle xwyry; 1/ \downarrow /- \rangle$.

The 8 possible FPs are compiled into one dynamic FFM, referred to as *Dynamic Disturb Coupling Fault*

Table 2. List of two-cell dynamic FFMs.

FFM	Fault primitives
dCFds	$\langle 0w0r0; 0/ \uparrow /- \rangle$, $\langle 0w1r1; 0/ \uparrow /- \rangle$, $\langle 1w0r0; 0/ \uparrow /- \rangle$, $\langle 1w1r1; 0/ \uparrow /- \rangle$, $\langle 0w0r0; 1/ \downarrow /- \rangle$, $\langle 0w1r1; 1/ \downarrow /- \rangle$, $\langle 1w0r0; 1/ \downarrow /- \rangle$, $\langle 1w1r1; 1/ \downarrow /- \rangle$
dCFrd	$\langle x; 0w0r0/ \uparrow /1 \rangle$, $\langle x; 0w1r1/ \downarrow /0 \rangle$, $\langle x; 1w0r0/ \uparrow /1 \rangle$, $\langle x; 1w1r1/ \downarrow /0 \rangle$
dCFdrd	$\langle x; 0w0r0/ \uparrow /0 \rangle$, $\langle x; 0w1r1/ \downarrow /1 \rangle$, $\langle x; 1w0r0/ \uparrow /0 \rangle$, $\langle x; 1w1r1/ \downarrow /1 \rangle$
dCFir	$\langle x; 0w0r0/0/1 \rangle$, $\langle x; 0w1r1/1/0 \rangle$, $\langle x; 1w0r0/0/1 \rangle$, $\langle x; 1w1r1/1/0 \rangle$

(dCFds): a write operation followed immediately by a read operation performed on the a-cell causes the v-cell to flip. The FPs of dCFds are given in the first block of Table 2.

3.2.2. Faults Caused by S_{vv}.
The FPs for two-cell dynamic faults, where the two operations are applied to the v-cell, are represented as:

$$\langle S_{vv}/F/R \rangle = \langle S_a; S_v/F/R \rangle_{a,v}$$

S_a describes the *state* of the a-cell, while S_v describes $xwyry$. Note that S_a does not specify an operation applied to the a-cell, but the *state* of the a-cell, since only S_{vv} is considered. Now it can be verified easily that there are 24 possible FPs:

- 12 FPs denoted as $\langle 0; S_v/F/R \rangle_{a,v}$, and
- 12 FPs as $\langle 1; S_v/F/R \rangle_{a,v}$

Note that in both representations, $\langle S_v/F/R \rangle$ represents the 12 possible single-cell dynamic FPs discussed in Section 3.1. The 24 possible FPs are compiled into three dynamic FFMs; they are given together with their FPs in Table 2; in the table 'x' denotes 0 or 1.

1. *Dynamic Read Destructive Coupling Fault (dCFrd)*: A write followed immediately by a read operation performed on the v-cell changes the data in the v-cell and returns an *incorrect* value on the output, if the a-cell is in a certain specific state. The dCFrd consists of eight FPs.
2. *Dynamic Deceptive Read Destructive Coupling Fault (dCFdrd)*: A write followed immediately by a read operation performed on the v-cell changes

the data in the v-cell and returns a *correct* value on the output, if the a-cell is in a certain specific state. The dCFdrd consists of eight FPs.

3. *Dynamic Incorrect Read Coupling Fault (dCFir)*: A read operation performed *immediately* after a write operation on the v-cell returns an *incorrect* value on the output, while the v-cell remains in its correct state, if the a-cell is in a certain specific state. The dCFir consists of eight FPs.

4. Importance of Dynamic Faults

This section will show the presence of dynamic faults in dynamic RAMs (DRAMs) as well as in static RAMs (SRAMs). First the validation of dynamic faults for DRAMs will be presented based on defect injection and circuit simulation; and thereafter for SRAMs based on the functional analysis of some failed test patterns during DPM screening measurements.

4.1. Validation of Dynamic Faults for DRAMs

In order to show the importance of dynamic faults, fault analysis based on defect injection and SPICE simulation has been performed. The defects are injected in the reduced electrical model of a DRAM, causing for example a (partial) open connections. This section describes the performed analysis and the acquired results [4].

Opens represent unwanted resistances on a signal line that is supposed to conduct perfectly. In this section, the simulation results of the open within a DRAM cell OC shown in Fig. 2 will be discussed; for other examples see [4]. The behavior of the DRAM is studied after injecting and simulating OC. The analysis considers open resistances within the range ($10\Omega \leq R_{op} \leq 10$ MΩ) on a logarithmic scale using 5 points per decade, in addition to $R_{op} = \infty$ Ω. Each injected

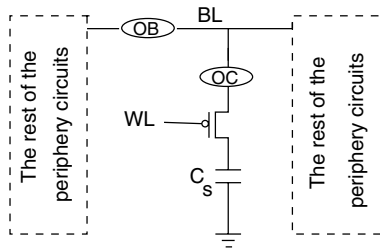open in the memory model creates floating nodes, the voltage of which is varied between V_{DD} and GND on a linear scale using 10 points.

For each value of the open resistance (R_{op}) and of the initial floating node voltage (U_{init}), the following operation sequences are performed and inspected for proper functionality: $0w0$, $0w1$, $1w0$, $1w1$, $0r0$, $1r1$, $0w0r0$, $0w1r1$, $1w0r0$ and $1w1r1$ (e.g., $0w1$ denotes a write 1 applied to a cell whose content is 0). As a result, the faulty behavior resulting from the analysis of opens is represented as regions in the (U_{init}, R_{op}) plane. Each region contains a number of sensitized FPs that describe the FFM of the memory in this region.

As an example, the results of the fault analysis performed on OC are given in Fig. 3, which shows the observed faulty behavior in the (U_{init}, R_{op}) plane. The figure shows a number of different fault regions for different combinations of U_{init} and R_{op}. In the figure, the static and dynamic sensitized faults are listed:

- Static Faults [9, 24]:
  $TF_d = \langle 1w0/1/-\rangle$: a down transition fault.
  $TF_u = \langle 0w1/0/-\rangle$: an up transition fault.
  $IRF_0 = \langle 0r0/0/1\rangle$: incorrect read 0 fault.
  $RDF_0 = \langle 0r0/\uparrow/1\rangle$: read 0 destructive fault.
  $WDF_0 = \langle 0w0/\uparrow/-\rangle$: write 0 disturb fault.
- Dynamic Faults (see Table 1):
  $dRDF_{00} = \langle 0w0r0/\uparrow/1\rangle$.
  $dRDF_{10} = \langle 1w0r0/\uparrow/1\rangle$.
  $dDRDF_{00} = \langle 0w0r0/\uparrow/0\rangle$.
  $dIRF_{00} = \langle 0w0r0/0/1\rangle$.

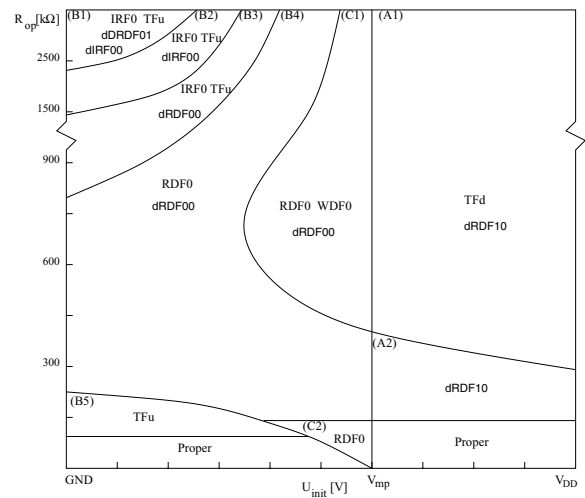*Fig. 2.* The opens OC and OB.

Fig. 3. Analysis results for OC.

The fault regions of Fig. 3 may be classified according to the initial floating node voltage under which they can be observed as follows.

A. Faults observed with $U_{init} = V_{DD}$: They consist of fault regions A1 and A2.
B. Faults observed with U_{init} = GND: They consist of fault regions B1, B2, B3, B4 and B5.
C. Faults only observed with GND $< U_{init} < V_{DD}$: They consist of fault regions C1 and C2.

Inspecting the faulty behavior shown in the figure reveals that, as a result of the open OC alone, almost all simulated dynamic sequences fail. This takes place in separate fault regions with their own initial voltage and defect resistance values. The fault region A2 only contains dRDF$_{10}$, which means that $1w0r0$ is the only failing S in this region. This, in turn, means that performing the traditional *static* analysis on this fault region reveals no improper memory behavior. Only by applying a *dynamic* sensitizing operation sequence is it possible to detect this improper behavior. This shows the significance of performing the dynamic analysis on memory devices.

A similar analysis has been done for the open OB (see Fig. 2). The results show that in some fault regions a dynamic fault behavior takes place in the absence of a static fault behavior. Table 3 shows which dynamic sequences fail for OC and OB [4]. The table shows that for OC most, but not all, dynamic sequences fail. However, for OB all simulated dynamic sequences cause a fault.

4.2.   *Validation of Dynamic Faults for SRAMs*

In an experiment at Intel, SRAM chips failing to pass the used test set have been analyzed in more detail. One of the surprising observations is that PMOVI [7] detects faults that can not be detected with any other test; e.g., March C- [17, 23]. Fig. 4 shows partially how PMOVI detects some faults that march C- does not detect; the operations printed in bold in the figure

Table 3.   Dynamic faults caused by OC and OB.

Open	Dynamic faulty behavior
OC	All FPs of dRDF (see Table 1)
	Two dDRDF FPs: $\langle 0w1r1/ \downarrow /1 \rangle$, $\langle 1w0r0/ \uparrow /0 \rangle$
	Two dIRF FPs: $\langle 0w0r0/0/1 \rangle$, $\langle 1w1r1/1/0 \rangle$
OB	All FPs of Table 1

PMOVI	$\{\Downarrow (w0); \Uparrow (r0, w1, \mathbf{r1}); \Uparrow (\mathbf{r1}, w0, r0); ...\}$
March C-	$\{\Updownarrow (w0); \Uparrow (r0, w1); \Uparrow (r1, w0); \Downarrow (r0, w1); ...\}$

Fig. 4.   Observation of dynamic faults in SRAMs.

are the operations detecting the unique faults. PMOVI detects some unique faults by the first 'r1' (printed in bold) that *immediately* follows a write operation. These faults are *not* detected by March C-, even though March C- consists of the same operations, but they belong to separate march elements and therefore the read is not applied immediately after the write.

It is clear from the above, that the detected fault require the application of a *read immediately after a write*, otherwise the fault will be not sensitized. This is a dynamic read destructive fault dRDF.

PMOVI detects also some unique faults by the second 'r1' (printed in bold) that were not detected by March C-. Such faults were sensitized by read-after-write (i.e., $0w1r1$), but not observed by that read since the read returned the expected value although the cell was flipped. These faults are the dynamic deceptive read destructive faults.

5.   Testing Dynamic Faults

In Section 4 the existence of dynamic faults has been validated based on SPICE simulation. In Section 3, a systematic way to develop fault models for dynamic faults has been introduced. In this section, the conventional memory tests will be analyzed for their capability of detecting dynamic faults. However, first the march notation will be introduced.

5.1.   *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 an increasing ($\Uparrow$) or a decreasing ($\Downarrow$) address order. When the address order is not relevant, the symbol $\Updownarrow$ is be used.

5.2.   *Effectiveness of Conventional Tests*

Table 4 gives the fault coverage of several well-known memory tests regarding the proposed dynamic FFMs.

Table 4. Dynamic fault coverage for different memory tests.

No.	Tests	T.L.	dRDF	dDRDF	dIRF	dCFds	dCFrd	dCFdrd	dCFir	Total FC	FC (%)
1	SCAN	4n	0/4	0/4	0/4	0/16	0/16	0/16	0/16	0/76	0
2	MATS+	5n	0/4	0/4	0/4	0/16	0/16	0/16	0/16	0/76	0
3	MATS++	6n	1/4	0/4	1/4	1/16	2/16	0/16	2/16	7/76	9.21
4	March C-	10n	0/4	0/4	0/4	0/16	0/16	0/16	0/16	0/76	0
5	PMOVI	13n	2/4	2/4	2/4	7/16	8/16	6/16	8/16	35/76	46.05
6	March U	13n	2/4	0/4	2/4	4/16	4/16	0/16	4/16	16/76	21.05
7	March SR	14n	2/4	0/4	2/4	4/16	4/16	0/16	4/16	16/76	21.05
8	March LR	14n	2/4	0/4	2/4	4/16	4/16	0/16	4/16	16/76	21.05
9	March B	17n	2/4	0/4	2/4	4/16	4/16	0/16	4/16	16/76	21.05
10	March LA	22n	2/4	2/4	2/4	8/16	8/16	8/16	8/16	38/76	50.00

The test length (T.L.) of each test is also included. The included march tests in the table are:

1. SCAN [1]: {⇑ $(w0)$; ⇑ $(r0)$; ⇑ $(w1)$; ⇑ $(r0)$}
2. MATS+ [19]: {⇕ $(w0)$; ⇑ $(r0, w1)$; ⇓ $(r1, w0)$}
3. MATS++ [6]: {⇕ $(w0)$; ⇑ $(r0, w1)$; ⇓ $(r1, w0, r0)$}
4. March C- [17, 23]: {⇕ $(w0)$; ⇑ $(r0, w1)$; ⇑ $(r1, w0)$; ⇓ $(r0, w1)$; ⇓ $(r1, w0)$; ⇕ $(r0)$}
5. PMOVI [7]: {⇓ $(w0)$; ⇑ $(r0, w1, r1)$; ⇑ $(r1, w0, r0)$; ⇓ $(r0, w1, r1)$; ⇓ $(r1, w0, r0)$}
6. March U [27]: {⇕ $(w0)$; ⇑ $(r0, w1, r1, w0)$; ⇑ $(r0, w1)$; ⇓ $(r1, w0, r0, w1)$; ⇓ $(r1, w0)$}
7. March SR [9]: {⇓ $(w0)$; ⇑ $(r0, w1, r1, w0)$; ⇑ $(r0, r0)$; ⇑ $(w1)$; ⇓ $(r1, w0, r0, w1)$; ⇓ $(r1, r1)$}
8. March LR [26]: {⇕ $(w0)$; ⇓ $(r0, w1)$; ⇑ $(r1, w0, r0, w1)$; ⇑ $(r1, w0)$; ⇑ $(r0, w1, r1, w0)$; ⇑ $(r0)$}
9. March B [22]: {⇕ $(w0)$; ⇑ $(r0, w1, r1, w0, r0, w1)$; ⇑ $(r1, w0, w1)$; ⇓ $(r1, w0, w1, w0)$; ⇓ $(r0, w1, w0)$}
10. March LA [28]: {⇕ $(w0)$; ⇑ $(r0, w1, w0, w1, r1)$; ⇑ $(r1, w0, w1, w0, r0)$; ⇓ $(r0, w1, w0, w1, r1)$; ⇓ $(r1, w0, w1, w0, r0)$; ⇓ $(r0)$}

In the table, e.g., "a/b" denotes that the test detects 'a' of the 'b' FPs of the correspondent FFM. E.g., MATS++ detects one of the four FPs of dRDF, while March B detects two 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 dCFds consists of 8 FPs (see Table 2); considering the position of the a-cell against the v-cell leads to 16 sub-FPs. E.g., March B detects 4 of 16 dCFds sub-FPs.

The table shows clearly that conventional memory tests constructed to detect the static faulty behavior of a specific defect, do not necessarily detect its dynamic faulty behavior. None of the tests of the table can cover the considered dynamic faults; even all of them together cannot achieve 100% fault coverage of the targeted faults. In addition, a relative comparison of the fault coverage of the tests shows that March LA and PMOVI are the tests with the best coverage. The fact the the conventional tests do not cover dynamic faults calls for the introduction of new test.

6. Tests for Dynamic Faults

Dynamic faults are divided into faults involving a single cell and faults involving two cells. March tests for each of these two subclasses will be introduced separately.

6.1. Test for Single-Cell Dynamic FFMs

March RAW1 ('read-after-write') given in Fig. 5 detects all single-cell dynamic FFMs of Table 1, which are based on 'read-after-write'.

March RAW1 has a test length of 13n including the initialization. Table 5 shows by which march elements (i.e., M_0 through M_8) of March RAW1, each FP belonging to each single-cell dynamic FFM, is sensitized and detected. In the table, 'S' stands for 'sensitizing', while 'D' stands for 'detecting'. Note that the test performs at most 2 operations per march element, which is

{ ⇕$(w0)$; ⇕$(w0, r0)$; ⇕$(r0)$; ⇕$(w1, r1)$; ⇕$(r1)$;
  M_0 M_1 M_2 M_3 M_4
⇕$(w1, r1)$; ⇕$(r1)$; ⇕$(w0, r0)$; ⇕$(r0)$ }
  M_5 M_6 M_7 M_8

Fig. 5. March RAW1.

Table 5. March RAW1 fault coverage.

FFM	FP	S	D
dRDF	$\langle 0w0r0/\uparrow/1\rangle$	M_1	M_1, M_2
	$\langle 0w1r1/\downarrow/0\rangle$	M_3	M_3, M_4
	$\langle 1w0r0/\uparrow/1\rangle$	M_7	M_7, M_8
	$\langle 1w1r1/\downarrow/0\rangle$	M_5	M_5, M_6
dDRDF	$\langle 0w0r0/\uparrow/0\rangle$	M_1	M_2
	$\langle 0w1r1/\downarrow/1\rangle$	M_3	M_4
	$\langle 1w0r0/\uparrow/0\rangle$	M_7	M_8
	$\langle 1w1r1/\downarrow/1\rangle$	M_5	M_6
dIRF	$\langle 0w0r0/0/1\rangle$	M_1	M_1, M_2
	$\langle 0w1r1/1/0\rangle$	M_3	M_3, M_4
	$\langle 1w0r0/0/1\rangle$	M_7	M_7, M_8
	$\langle 1w1r1/1/0\rangle$	M_5	M_5, M_6

Table 6. March RAW fault coverage.

FFM	FP	$v > a$	$v < a$
dCFds	$\langle 0w0r0;0/\uparrow/-\rangle$	M_1/M_1	M_3/M_3
	$\langle 0w1r1;0/\uparrow/-\rangle$	M_1/M_1	M_3/M_3
	$\langle 1w1r1;0/\uparrow/-\rangle$	M_4/M_5	M_2/M_3
	$\langle 1w0r0;0/\uparrow/-\rangle$	M_4/M_5	M_2/M_3
	$\langle 0w0r0;1/\downarrow/-\rangle$	M_3/M_4	M_1/M_2
	$\langle 0w1r1;1/\downarrow/-\rangle$	M_3/M_4	M_1/M_2
	$\langle 1w1r1;1/\downarrow/-\rangle$	M_2/M_2	M_4/M_4
	$\langle 1w0r0;1/\downarrow/-\rangle$	M_2/M_2	M_4/M_4
dCFdrd	$\langle 0;0w0r0/\uparrow/0\rangle$	M_3/M_3	M_1/M_1
	$\langle 0;0w1r1/\downarrow/1\rangle$	M_3/M_4	M_1/M_2
	$\langle 0;1w1r1/\downarrow/1\rangle$	M_2/M_2	M_4/M_4
	$\langle 0;1w0r0/\uparrow/0\rangle$	M_2/M_3	M_4/M_5
	$\langle 1;0w0r0/\uparrow/0\rangle$	M_1/M_1	M_3/M_3
	$\langle 1;0w1r1/\downarrow/1\rangle$	M_1/M_2	M_3/M_4
	$\langle 1;1w1r1/\downarrow/1\rangle$	M_4/M_4	M_2/M_2
	$\langle 1;1w0r0/\uparrow/0\rangle$	M_4/M_5	M_2/M_3

important to restrict the sensitized faults to 2-operation dynamic.

6.2. Test for Two-Cell Dynamic FFMs

A test to detect all two-cell dynamic FFMs of Table 2 is given in Fig. 6, and called *March RAW*. March RAW has a test length of $26n$ including the initialization. Note that the fault coverage is not limited to two-operation dynamic faults; the test also detects some dynamic faults where $\#O > 2$. Table 6 shows the march elements of March RAW responsible for sensitizing and detecting each of the faults listed in Table 2. In the table, a distinction is made between two cases: (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 $\langle 1w1r1;0/\uparrow/-\rangle$ is sensitized by M_4 and detected by M_5 when $v > a$; and it is sensitized by M_2 and detected by M_3 when $v < a$. Note that dCFrd and dCFir are not included in the table; the reader can easily verify that these faults

are also covered by March RAW, and that any test detecting dCFdrd will also detect dCFrd and dCFir. Note additionally that March RAW also detects all FFM1s of Table 1; therefore only March RAW can be used to *detect* all dynamic faults introduced in this paper. If the purpose is to *diagnose* the FFM1s, then March RAW1 has to be used.

Going back to Fig. 1 which shows the scope of the paper, the question that may be posed is whether March RAW detects simple static faults. It can be shown that March RAW also covers simple static faults. A detail analysis of such faults together with an optimal test is presented in [12]. The test referred as March SS (i.e., Simple Static) with a test length of $22n$ can be considered as a short version of March RAW.

7. Conclusion

In this paper the difference between static and dynamic faults in memories has been discussed; dynamic faults can take place in the absence of static faults as has been shown experimentally. An analytical approach for establishing fault models for dynamic faults has been presented. In addition, a set of fault models (under certain restrictions) has been introduced. The evaluation of the existing conventional memory tests shows that these tests cannot cover these fault models. Therefore,

```
{  ⇕ (w0) ;
       M₀
   ⇑ (r0, w0, r0, r0, w1, r1)  ;   ⇑ (r1, w1, r1, r1, w0, r0) ;
              M₁                              M₂
   ⇓ (r0, w0, r0, r0, w1, r1)  ;   ⇓ (r1, w1, r1, r1, w0, r0) ;
              M₃                              M₄
   ⇕ (r0)   }
     M₅
```

Fig. 6. March RAW.

two new memory tests have been derived to target such specific faults.

The fault models introduced for dynamic faults have been restricted to faults sensitized by 'a write followed immediately with a read'. These faults have been observed in DRAMs as well as in SRAMs. The question that arises now is whether other sequential operations can also sensitize dynamic faults; e.g., 'a write followed immediately by a write', 'a read followed immediately by a read', etc. The widely used 'hammer tests' (i.e., repeat a write or a read operation sequentially) may indicate the existence of such dynamic faults. Furthermore, the 'Holey Shmoo problem' [14] in which the L1 cache of IBM System/390 G6 microprocessor fails to pass consecutive write patterns also indicate that dynamic faults can be caused by 'a write followed immediately by another write'. It is clear from the above, that the set of fault models for dynamic faults has to be explored, and the appropriate test algorithms have to be established.

Note

1. The analysis of S_{va} and S_{av} will be the subject of an upcoming paper.

References

1. M.S. Abadir and J.K. Reghbati, "Functional Testing of Semiconductor Random Access Memories," *ACM Computer Surveys*, vol. 15, no. 3, pp. 175–198, 1983.
2. 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.
3. Z. Al-Ars and Ad J. van de Goor, "Impact of Memory Cell Array Bridges on the Faulty Behavior in Embedded DRAMs," in *Proc. of Asian Test Symposium*, 2000, pp. 282–289.
4. Z. Al-Ars and 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*, 2001, pp. 496–503.
5. Z. Al-Ars and A.J. van de Goor, "Approximating Infinite Dynamic Behavior for DRAM Cell Defects," in *Proc. IEEE VLSI Test Symp.*, 2002, pp. 401–406.
6. M.A. Breuer and A.D. Friedman, *Diagnosis and Reliable Design of Digital Systems*, Woodland Hills, CA: Computer Science Press, 1976.
7. J.H. De Jonge and A.J. Smeulders, "Moving Inversions Test Pattern is Thorough, Yet Speedy," in *Comp. Design*, 1976, pp. 169–173.
8. R. Dekker et al., "A Realistic Fault Model and Test Algorithms for Static Random Access Memories," *IEEE Trans. on Computers,* vol. C9, no. 6, pp. 567–572, 1990.
9. S. Hamdioui and A.J. van de Goor, "Experimental Analysis of Spot Defects in SRAMs: Realistic Fault Models and Tests," in *Proc. of Asian Test Symposium*, Taipei, Taiwan, 2000, pp. 131–138.
10. S. Hamdioui and A.J. van de Goor, "Thorough Testing of any Multi-Port Memory with Linear Tests," *In IEEE Trans. on CAD*, vol. 21, no. 2, pp. 217–231, Feb. 2002.
11. S. Hamdioui and A.J. van de Goor, "Efficient Tests for Realistic Faults in Dual-Port SRAMs," in *IEEE Trans. on Computers*, vol. 51, no. 5, pp. 460–473, May 2002.
12. S. Hamdioui, A.J. van de Goor, and M. Rodgers, "March SS: A Test for All Static Simple RAM Faults," in *Proc. of IEEE International Workshop on Memory Technology, Design and Test*, 2002, pp. 95–100.
13. S. Hamdioui, Z. Al-ars, and A.J. van de Goor, "Testing Static and Dynamic Faults in Random Access Memories," in *Proc. of IEEE VLSI Test Symposium*, 2002, pp. 395–400.
14. W. Huott et al., "The Attack of the 'Holey Shmoos': A Case of the Advanced DFD and Picosecond Imaging Circuit Analysis (PICA)," in *Proc. of IEEE International Test Conference,* 1999, pp. 883–891.
15. M. Inoue et al., "A New Test Evaluation Chip for Lower Cost Memory Tests," *IEEE Design and Test of Computers*, vol. 10, no. 1, pp. 15–19, March 1993.
16. V.K. Kim and T. Chen, "On Comparing Functional Fault Coverage and Defect Coverage for Memory Testing," *IEEE Trans. on CAD*, vol. 18, no. 11, pp. 1676–1683, 1999.
17. M. Marinescu, "Simple and Efficient Algorithms for Functional RAM Testing," in *Proc. of IEEE International Test Conference*, 1982, pp. 236–239.
18. S. Naik et al., "Failure Analysis of High Density CMOS SRAMs," *IEEE Design and Test of Computers*, vol. 10, no. 1, pp. 13–23, June 1993.
19. 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.
20. C.A. Papachristou and N.B. Saghal, "An Improved Method for Detecting Functional Faults in Random Access Memories," *IEEE Trans. on Computers,* vol. C-34, no. 3, pp. 110–116, 1985.
21. 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*, 1999, pp. 983–992.
22. D.S. Suk and S.M. Reddy, "A March Test for Functional Faults in Semiconductor Random-Access Memories," *IEEE Trans. on Computers*, vol. C30, no. 12, pp. 982–985, 1981.
23. A.J. van de Goor, *Testing Semiconductor Memories, Theory and Practice*, ComTex Publishing, Gouda: The Netherlands, 1998. Web: http://cardit.et.tudelft.nl/v̄dgoor
24. A.J. van de Goor and Z. Al-Ars, "Functional Fault Models: A Formal Notation and Taxonomy," in *Proc. of IEEE VLSI Test Symposium*, 2000, pp. 281–289.
25. A.J. van de Goor and J. de Neef, "Industrial Evaluation of DRAMs Tests," in *Proc. of Design Automation and Test in Europe*, March 1999, pp. 623–630.
26. A.J. van de Goor and G. Gaydadjiev, "March LR: A Memory Test for Realistic Linked Faults," in *Proc. IEEE VLSI Test Symposium*, 1996, pp. 272–280.
27. 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*, vol. 144, no. 3, pp. 155–160, 1997.

28. A.J. van de Goor et al., "March LA: A Test for Linked Memory Faults," in *Proc. of European Design and Test Conf.*, 1999, p. 627.
29. P.K. Veenstra et al., "Testing of Random Access Memories: Theory and Practice," *IEE Proc. of Circuits, Devices, and Systems*, vol. 135, no. 1, pp. 24-28, 1988.

Said Hamdioui received his MSEE degree with honors form Delft University of Technology, in Delft, The Netherlands. Additionally, he received his Ph.D. degree with honors from the same university. Since then, he has been working at the same University. He interned with Intel Corporation, USA, and was responsible for developing new low cost and efficient test algorithms for advanced Intel singe-port and multi-port cache designs. Currently, he is a visiting faculty a Intel Santa Clara. His research interests concern fault modeling, failure analysis, fault simulation, memory testing, test design, DFT, etc. He has published numerous papers in the area of testing, and he is an associate member of the IEEE.

Zaid Al-Ars received his MS degree in electrical engineering with honors from the Delft University of Technology, the Netherlands in the year 2000. He is working toward his Ph.D. degree in electrical engineering with the same university in cooperation with Infineon Technologies, Munich, Germany, where he is currently based. His research project involves systematic fault analysis, and test generation and optimization for commodity as well as embedded DRAM products. He published over 10 papers in the field of electrical defect simulation, fault modeling and test generation in memory devices.

Ad van de Goor received his MSEE degree from Delft University of Technology, in Delft, The Netherlands. Additionally, he received the MSEE and Ph.D. degrees from Carnegie-Mellon University, Pennsylvania. He worked for the Digital Equipment Corporation in Maynard, Massachusetts, as the chief architect of the PDP-11/45 computer, and for IBM in The Netherlands and in USA, being responsible for architecture of the embedded systems. Currently, he is a professor of computer engineering at Delft University of Technology. His main research interest is testing logic and memories. He has written two books and more than 120 papers in the area of computer architecture and testing. He is a member of the editorial board of the *Journal of Electronic Testing: Theory and Applications*, and a IEEE fellow.

Mike Rodgers has been with Intel for 16 years since BSEE from U. of Illinois and additional studies at U. of Chicago. He has held various positions in Microprocessor Q&R and Test Technology, including running Q&R and FA groups in Santa Clara CA, Japan, and the Chandler, AZ Test Factory. He has worked with DF* teams on all IA CPUs going since the 286, was primary driver for Intel's cache DFT and test methods and now focuses on DFT and methods for manufacturing test. He currently runs Test Planning and Integration in Design Technology in Santa Clara, CA and is in charge of Technology Planning for Test between technology, product, and factory teams. He co-chairs the Test TWG of the ITRS Roadmap. He is a member of the IEEE and the TTTC and the author of numerous papers on test. He is an avid large format photographer and printer in his spare time.