Functional Memory Faults: A Formal Notation and a Taxonomy

Ad J. van de Goor   Zaid Al-Ars
Section CARDIT, Faculty of Information Technology and Systems
Delft University of Technology Mekelweg 4, 2628 CD Delft, The Netherlands
E-mail: vdGoor@cardit.et.tudelft.nl

Abstract: *This paper presents a notation for describing functional fault models, which may occur in memory devices. Using this notation, the space of all possible memory faults has been constructed. It has been shown that this space is infinite, and contains the currently established functional fault models. New fault models in this space have been identified and verified using resistive and capacitive defect injection and simulation of a DRAM model.*

Keywords: *Functional memory fault model, taxonomy, fault space, simulation stimuli*

1 Introduction

For test purposes, faults in memories are usually modeled as functional faults such that functional tests can be used to detect those faults. The advantage of functional tests is that completeness and irredundancy proofs become a logical problem, which is rather easy to deal with. In addition, the design of a functional test, given a (set of) functional fault model(s) can be done using a systematic set of rules, which has already been automated to some extent [vdGoor94, Zarrineh98]. From this, it is clear that the knowledge of the functional faults, which may occur in the memory under test, is essential for designing tests and for getting a low PPM level.

Many *functional fault models (FFMs)* for memories have been introduced in the past; some well known FFMs, which date back to before 1980, are [vdGoor98]: address decoder faults, stuck-at faults, inversion and idempotent coupling faults, and neighborhood pattern sensitive faults. Of later dates are the following FFMs: data retention faults [Dekker90], stuck-open faults [Dekker90], state coupling faults [Dekker90], read disturb faults [Adams96], deceptive read disturb faults [Adams96], and disturb coupling faults [vdGoor96]. The process of detecting new FFMs has been very ad-hoc and, therefore, slow. Experimental results of applying a large number of tests to a large number of chips [vdGoor99a, Schanstra99] indicate that many functional tests do detect faults in memories, which cannot be explained using the current set of known FFMs. This means that additional FFMs do exist.

Establishing new FFMs is usually done by inserting resistive defects into the electrical schematic of a memory, followed by SPICE simulation of the effect of the resistive defect. One problem with this method relates to the stimuli used to drive the simulator; some authors have used existing march tests to drive the simulator [Mak98] such that the fault coverage of the used march tests for the inserted defects can be established. However, this does not lead to new FFMs and new tests to detect faults of the new FFMs [vdGoor99b]. In order to show the presence/absence of a particular FFM, the correct sequence of stimuli for the simulator has to be used. These stimuli are FFM specific and hence, the FFMs to be anticipated have to be known in advance. If not, the detection of new FFMs is an ad-hoc activity.

The fact that many faults in memories do exist for which no FFM has been established yet, together with the fact that the FFMs to be anticipated have to be known in advance, calls for a systematic way to construct potential FFMs and explore the space of all potential FFMs.

This paper is organized as follows. Section 2 discusses the basic properties of memory faulty behavior and ends with a formal definition of an FFM. Section 3 introduces a taxonomy for the space of FFMs. Section 4 and Section 5 are concerned with single-cell and two-cell FFMs, respectively. Section 6 uses an electrical DRAM model to verify the defined FFMs based on defect injection and fault simulation. Finally, Section 7 ends with the conclusions.

2 Properties of fault models

Functional faults can be defined as the deviation of the observed memory behavior from the functionally specified one under a set of performed operations. Therefore, two basic ingredients can be identified to any FFM: (1) a list of performed memory operations, and (2) 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 (SOS)*[1]. The observed memory behavior that deviates from the expected one is called a *faulty behavior*. A general notation to represent operation sequences is given first, followed by a notation of the faulty behavior.

2.1 Representing operation sequences

In order to perform a memory operation, an address should first be provided to indicate the cell to be accessed. Secondly, an operation type should be selected to distinguish between a read (denoted by r) and a write (denoted by w) operation. Thirdly, in case of a write operation, the data to be written should also be provided.

Besides the above three items, it is known that the faulty behavior of a memory is also affected by the initial logic value present inside the cell. Based on these observations, any SOS can be represented by the following notation

$$c(iOd)_1\ c(iOd)_2\ ...\ c(iOd)_k\ ...\ c(iOd)_n$$

where c: cell address used by k-th operation,
i: initial stored value in cell c, $i \in \{0,1\}$,
O: type of operation on c, $O \in \{w,r\}$, and
d: data to be written into c, $d \in \{0,1\}$.

Notice that d is only needed if the performed operation is a write ($O = w$), however it is the custom to set d to the expected output if the operation is a read ($O = r$). For example, if an operation sequence is denoted by $3(0w1)\ 3(1r1)\ 2(0r0)$ then the sequence starts with accessing cell 3, which contains a 0, and writing a 1 into it, then reading the written 1 from cell 3, and finally reading a stored 0 from cell 2.

[1]In the literature, a distinction is made between operations that sensitize the fault internally and those that result in detecting the fault on output lines. Since, in this paper, we are mainly concerned with the internal memory behavior, this distinction is not necessary.

Sometimes a fault is sensitized without the need to perform any operation. This way, simply setting the cell into a known initial state is enough to sensitize the fault. This situation can be described by replacing $c(iOd)$ with $c(i)$. For example, observing the state of cell 2, which contains a 0, without accessing it can be denoted by $2(0)$. Here, we assume to have the means to initialize the cell to a known initial state without accessing it. Generally, this is not possible externally, but this paper is mainly concerned with the internal memory behavior which makes it possible to know and observe the initial state of the cell without accessing it.

2.2 Describing faulty behavior

Throughout the 1980s and during the first half of the 1990s, the only functional parameter considered relevant to the faulty behavior was the stored logic state in the memory cell [vdGoor98]. Recently, another functional parameter, *the output value of a read operation*, has also been considered to be relevant [vdGoor99b]. Therefore, any difference between the observed and expected memory behavior can be denoted by the following notation $<S/F/R>$. S describes the SOS, F describes the state stored in the faulty cell, $F \in \{0,1\}$, and R describes the logic output of a read operation, $R \in \{0,1,-\}$. $R = -$ is used in case a *write operation*, and not a read, is the operation that sensitizes the fault. The difference between the observed and expected memory behavior denoted by $<S/F/R>$ is referred to as a *fault primitive* (FP). The notion of FPs makes it possible to give a precise definition of an FFM as understood for memory devices. This definition is presented next.

A **functional fault model** is a non-empty set of fault primitives.

3 Space of all memory FPs

FPs can be classified according to $\#C$, the number of different cells accessed during an SOS, and according to $\#O$, the number of different operations performed in an SOS (see Figure 1). For example, if the SOS is $3(0w1)\ 3(1r1)\ 2(0r0)$ then $\#C = 2$, since two different cells (cell 2 and cell 3) are accessed by this sequence. On the other hand, $\#O = 3$ for this SOS since cell 2 is accessed once and cell 3 is accessed twice.

Depending on $\#C$, FPs can be divided into the following classes:

- If $\#C = 1$ then the FP sensitized by the corresponding SOS is called a *single-cell FP*.
- If $\#C > 1$ then the FP sensitized by the corresponding SOS is called a *coupling FP*. If $\#C = 2$ then it is described as *two-coupling FP* or *two-cell FP*. If $\#C = 3$ then it is described as *3-coupling FP*, etc.

Depending on $\#O$, FPs can be divided into the following classes:

- If $\#O \leq 1$ then the FP sensitized by the corresponding SOS is called a *static FP*.
- If $\#O > 1$ then the FP sensitized by the corresponding SOS is called a *dynamic FP*. If $\#O = 2$ then it is described as *2-operation dynamic FP*. If $\#O = 3$ then it is described as *3-operation dynamic FP*, etc.

Figure 1 shows a taxonomy of the space of FPs. It is important to note that the two ways to classify FPs are independent, since their definition is based on independent factors of the SOS. As a result, a single-cell FP can be static, or dynamic with any number of operations. The same applies to coupling FPs.

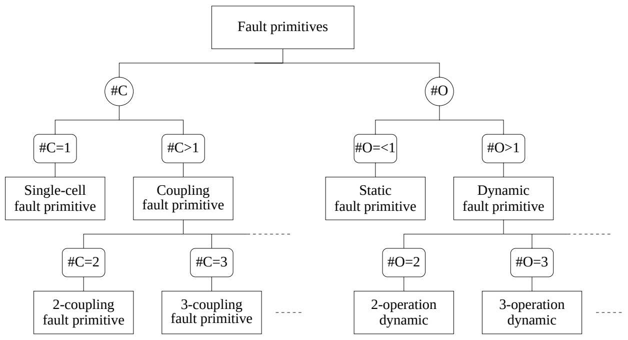

Figure 1. Taxonomy of fault primitives.

Since an FFM is defined as a set of FPs, it is expected that FFMs will inherit the properties of FPs. For example, if an FFM is defined as a collection of single-cell FPs, then the FFM is a single-cell fault model, etc. If an FFM consists of FPs classified into inconsistent classes, single-cell and two-cell FPs for example, it is described as a single-cell and a two-cell fault model.

The taxonomy above can be extended to include linked faults [vdGoor98] and data retention faults [Dekker90].

4 Single-cell FFMs

Single-cell FFMs are those fault models that consist of single-cell FPs (i.e., they are FFMs that describe an incorrect behavior of either the data stored in a single cell or the operations performed on it). For single-cell FFMs, the general notation for the sensitizing operation, $c(iOd)_1$ ... $c(iOd)_k$ ... $c(iOd)_n$ can be simplified to $(iOd)_1$ ... $(iOd)_k$ ... $(iOd)_n$ performed on cell c, because all operations are applied to a single cell with address c. Moreover, the expected initial value of the cell (i) before performing the k-th operation is always the same as the stored value in the cell (d) after performing the (k–1)-th operation. Therefore, it is possible to remove all i values except the first one, resulting in the following simplified notation: $i(Od)_1(Od)_2...(Od)_n$ performed on cell c. For example, the SOS $4(1w0)\ 4(0w1)\ 4(1r1)$, which is performed on a single cell, can be simplified to: $1w0w1r1$ performed on cell 4. Single-cell FFMs can be classified into static FFMs and dynamic FFMs.

Table 1. All possible combinations of the values in the $<S/F/R>$ notation resulting in single-cell static FPs.

#	S	F	R	FP	Fault model
1	0	1	–	$< 0/1/- >$	SF_0
2	1	0	–	$< 1/0/- >$	SF_1
3	$0w0$	1	–	$< 0w0/1/- >$	WDF_0
4	$0w1$	0	–	$< 0w1/0/- >$	$TF\uparrow$
5	$1w0$	1	–	$< 1w0/1/- >$	$TF\downarrow$
6	$1w1$	0	–	$< 1w1/0/- >$	WDF_1
7	$0r0$	0	1	$< 0r0/0/1 >$	IRF_0
8	$0r0$	1	0	$< 0r0/1/0 >$	$DRDF_0$
9	$0r0$	1	1	$< 0r0/1/1 >$	RDF_0
10	$1r1$	0	0	$< 1r1/0/0 >$	RDF_1
11	$1r1$	0	1	$< 1r1/0/1 >$	$DRDF_1$
12	$1r1$	1	0	$< 1r1/1/0 >$	IRF_1

4.1 Static single-cell FFMs

Static single-cell FFMs describe faults sensitized by performing at most one operation on the faulty cell (i.e., $\#O \leq 1$). As mentioned earlier, a particular FP is denoted by $<S/F/R>$. $S \in \{0, 1, 0w0, 0w1, 1w0, 1w1, 0r0, 1r1\}$ for static FPs, $F \in \{0, 1\}$ while $R \in \{0, 1, -\}$.

Now that the possible values for S, F and R are defined for single-cell static FPs, it is possible to list all detectable FPs using this notation. Table 1 lists all possible combinations of the values, in the $<S/F/R>$ notation, that result in FPs. The remaining combinations of the S, F and R values do not represent a faulty behavior. For example, $<0w0/0/->$ corresponds to a correct $w0$ operation after which the cell contains a 0, as expected.

The way the 12 FPs of Table 1 relate to FFMs is shown in the last column of the table. Below, a list is compiled of a number of well known and new single-cell static FFMs, described in terms of non-empty sets of FPs.

1. **State faults (SF$_x$)**—A cell is said to have an SF if the logic value of the cell flips before it is accessed, even if no operation is performed on it[2]. Two types of SF exist: SF$_0$ = {<0/1/−>}, with FP #1, and SF$_1$ = {<1/0/−>}, with FP #2.
2. **Transition faults (TFx)**—A cell is said to have a TF if it fails to undergo a transition ($0 \rightarrow 1$ or $1 \rightarrow 0$) when it is written.
3. **Read disturb faults (RDF$_x$)** [Adams96]—A cell is said to have an RDF if a read operation performed on the cell changes the data in the cell and returns an incorrect value on the output.
4. **Write disturb faults (WDF$_x$)**—A cell is said to have a WDF if a non-transition write operation ($0w0$ or $1w1$) causes a transition in the cell.
5. **Incorrect read faults (IRF$_x$)**—A cell is said to have an IRF if a read operation performed on the cell returns the incorrect logic value, while keeping the correct stored value in the cell.
6. **Deceptive read disturb faults (DRDF$_x$)** [Adams96]—A cell is said to have a DRDF if a read operation performed on the cell returns the correct logic value, while it results in changing the contents of the cell.
7. **Stuck-at faults (SAF$_x$)**—A cell is said to have a SAF if it remains always stuck at a given value for all performed operations. Two types of SAF exist: SAF$_0$ = {<$\forall$/0/−>}, and SAF$_1$ = {<$\forall$/1/−>}.

   $\forall$ symbolizes the idea that for all operations the same value remains in the cell. Therefore, $S = \forall$ can be replaced by only those operations that sensitize the fault. This leads to the following equivalent SAF definitions. SAF$_0$ = {<1/0/−>, <$0w1$/0/−>, <$1w1$/0/−>} = SF$_1$ ∪ TF↑ ∪ WDF$_1$, and SAF$_1$ = {<0/1/−>, <$1w0$/1/−>, <$0w0$/1/−>} = SF$_0$ ∪ TF↓ ∪ WDF$_0$. The ∪ sign is the usual mathematical union sign. In terms of FPs, a ∪ connecting a number of sets with FPs means that the FPs are all present in the faulty behavior simultaneously. That is, performing each SOS results in sensitizing the corresponding FP[3].

Table 2. All possible combinations of the values in the $<S/F/R>$ notation resulting in a single-cell 2-operation FP.

#	S	F	R	FP	#	S	F	R	FP
1	$0w0w0$	1	−	$< 0w0w0/1/- >$	2	$0w0w1$	0	−	$< 0w0w1/0/- >$
3	$0w0r0$	0	1	$< 0w0r0/0/1 >$	4	$0w0r0$	1	0	$< 0w0r0/1/0 >$
5	$0w0r0$	1	1	$< 0w0r0/1/1 >$					
6	$0w1w0$	1	−	$< 0w1w0/1/- >$	7	$0w1w1$	0	−	$< 0w1w1/0/- >$
8	$0w1r1$	0	0	$< 0w1r1/0/0 >$	9	$0w1r1$	0	1	$< 0w1r1/0/1 >$
10	$0w1r1$	1	0	$< 0w1r1/1/0 >$					
11	$1w0w0$	1	−	$< 1w0w0/1/- >$	12	$1w0w1$	0	−	$< 1w0w1/0/- >$
13	$1w0r0$	0	1	$< 1w0r0/0/1 >$	14	$1w0r0$	1	0	$< 1w0r0/1/0 >$
15	$1w0r0$	1	1	$< 1w0r0/1/1 >$					
16	$1w1w0$	1	−	$< 1w1w0/0/- >$	17	$1w1w1$	0	−	$< 1w1w1/0/- >$
18	$1w1r1$	0	0	$< 1w1r1/0/0 >$	19	$1w1r1$	0	1	$< 1w1r1/0/1 >$
20	$1w1r1$	1	0	$< 1w1r1/1/0 >$					
21	$0r0w0$	1	−	$< 0r0w0/1/- >$	22	$0r0w1$	0	−	$< 0r0w1/0/- >$
23	$0r0r0$	0	1	$< 0r0r0/0/1 >$	24	$0r0r0$	1	0	$< 0r0r0/1/0 >$
25	$0r0r0$	1	1	$< 0r0r0/1/1 >$					
26	$1r1w0$	1	−	$< 1r1w0/1/- >$	27	$1r1w1$	0	−	$< 1r1w1/0/- >$
28	$1r1r1$	0	0	$< 1r1r1/0/0 >$	29	$1r1r1$	0	1	$< 1r1r1/0/1 >$
30	$1r1r1$	1	0	$< 1r1r1/1/0 >$					

The FFMs defined in the first 6 items of the list above cover the space of all 12 single-cell static FPs. Any single-cell static FFM can be represented as the union set of two or more of these 12 FPs. For example, the SAF$_1$ has been defined above as a the union set of 3 FPs. Another example, if a defect results in a faulty behavior represented by a RDF$_1$ and a WDF$_1$, then the corresponding behavior is described as {<$1r1$/0/0>} ∪ {<$1w1$/0/−>} = RDF$_1$ ∪ WDF$_1$.

4.2 Dynamic single-cell FFMs

Dynamic FFMs are faults sensitized with more than one operation (i.e., $\#O > 1$), and as such, there is an infinite number of them. Therefore, we restrict ourselves to the case with $\#O = 2$.

Any particular FP is denoted by $< S/F/R >$, where $S \in \{i(Od)_1(Od)_2 : O \in \{r, w\}, i \in \{0,1\}$ and $d \in \{0,1\}\}$ for 2-operation dynamic FPs, $F \in \{0,1\}$, and $R \in \{0,1,-\}$. Based on the values of S, F and R, all 30 detectable single-cell 2-operation dynamic FPs are compiled in Table 2.

Again the question arises of how the 30 FPs of Table 2 relate to specific FFMs. Each FP can be used to define a corresponding FFM resulting in 30 dynamic FFMs. Below, not all these FFMs are presented since we will only attempt to verify a limited number of dynamic FFMs (see Section 6). The names of the FFMs are chosen in such a way that they represent an extension of the existing single-cell static FFMs.

[2]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.

[3]To be precise, stuck-at faults are not strictly static FFMs. SAFs are very general FFMs that describe general dynamic faulty behavior and, therefore, can include many (static and dynamic) single-cell FPs.

1. **Dynamic read disturb faults ($\mathbf{RDF}_{xy}$)**—The dynamic RDF has a similar definition to the RDF in the static case. It is a fault whereby an $xwyry$ SOS changes the stored logic value to $\overline{y}$ and gives an incorrect output. Four types of dynamic RDF exist: $\text{RDF}_{00} = \{<0w0r0/1/1>\}$, with FP #5, $\text{RDF}_{11} = \{<1w1r1/0/0>\}$, with FP #18, $\text{RDF}_{01} = \{<0w1r1/0/0>\}$, with FP #8, and $\text{RDF}_{10} = \{<1w0r0/1/1>\}$, with FP #15.
2. **Dynamic incorrect read faults ($\mathbf{IRF}_{xy}$)**—The dynamic IRF has a similar definition to the IRF in the static case. It is a fault whereby an $xwyry$ SOS returns the logic value $\overline{y}$ while keeping the correct state of the cell. Four types of dynamic IRF exist: $\text{IRF}_{00} = \{<0w0r0/0/1>\}$, with FP #3, $\text{IRF}_{11} = \{<1w1r1/1/0>\}$, with FP #20, $\text{IRF}_{01} = \{<0w1r1/1/0>\}$, with FP #10, and $\text{IRF}_{10} = \{<1w0r0/0/1>\}$, with FP #13.
3. **Dynamic deceptive read disturb faults ($\mathbf{DRDF}_{xy}$)**—The dynamic DRDF has a similar definition to the DRDF in the static case. It is a fault whereby an $xwyry$ SOS returns the correct logic value y while destroying the state of the cell. Four types of dynamic DRDF exist: $\text{DRDF}_{00} = \{<0w0r0/1/0>\}$, with FP #4, $\text{DRDF}_{11} = \{<1w1r1/0/1>\}$, with FP #19, $\text{DRDF}_{01} = \{<0w1r1/0/1>\}$, with FP #9, and $\text{DRDF}_{10} = \{<1w0r0/1/0>\}$, with FP #14.

5 Two-cell FFMs

Two-cell FFMs are faults consisting of two-cell FPs. The cell that shows the faulty behavior is called the *victim*, while the cell with which the victim cell interacts to produce the fault is called the *aggressor*. Two-cell FFMs can be classified into static FFMs and dynamic FFMs.

5.1 Static two-cell FFMs

Static two-cell FFMs describe faults sensitized by performing at most one operation while considering the effect two different cells have on each other, such that $S = c(iOd)_1 \ldots c(iOd)_k \ldots c(iOd)_n$ can be simplified to one of the following sequences:

$$\begin{aligned} S_{sn} &= a(i)\ v(j)\text{: no cell accessed} \\ S_{sa} &= a(iOd)\ v(j)\text{: only aggressor accessed} \\ S_{sv} &= a(i)\ v(jOd)\text{: only victim accessed} \end{aligned}$$

where i and j are the initial states of the aggressor and victim, respectively, and a and v stand for the addresses of the aggressor and the victim, respectively. The subscript s indicates a static sequence; the subscript n in S_{sn} indicates that it represents a sequence with *no* performed operations and that the states of the two cells are only observed; and the subscripts a in S_{sa} and v in S_{sv} indicate that there is only one operation performed on the aggressor and the victim, respectively. Note that the operation sequence part concerning the aggressor is put always first. For example, 2(1) 3(0) means that cell 2 is set to 1 and taken as aggressor, while cell 3 is set to 0 and observed as a victim for a possible change of state. Usually, two-cell sequences are given a special notation that separates the sequence part concerned with the aggressor from the sequence part concerned with the victim, and mentions the addresses of the cells afterwards in the order a, v. As a result, any two-cell static operation sequence S can be presented as follows: $S = S_a; S_v$ performed on cells a, v. Using this special notation, the sequence $S = 2(1)\ 3(0)$ is transformed into $S = 1; 0$ performed on cells 2, 3, where $S_a = 1$, $S_v = 0$, $a = 2$ and $v = 3$.

Table 3. All possible combinations of the values in the $<S/F/R>$ notation resulting in two-cell static FPs.

#	S_a	S_v	F	R	$< S_a; S_v/F/R >$	#	S_a	S_v	F	R	$< S_a; S_v/F/R >$
1	0	0	1	–	$< 0; 0/1/- >$	2	0	1	0	–	$< 0; 1/0/- >$
3	1	0	1	–	$< 1; 0/1/- >$	4	1	1	0	–	$< 1; 1/0/- >$
5	$0w0$	0	1	–	$< 0w0; 0/1/- >$	6	$0w0$	1	0	–	$< 0w0; 1/0/- >$
7	$0w1$	0	1	–	$< 0w1; 0/1/- >$	8	$0w1$	1	0	–	$< 0w1; 1/0/- >$
9	$1w0$	0	1	–	$< 1w0; 0/1/- >$	10	$1w0$	1	0	–	$< 1w0; 1/0/- >$
11	$1w1$	0	1	–	$< 1w1; 0/1/- >$	12	$1w1$	1	0	–	$< 1w1; 1/0/- >$
13	$0r0$	0	1	–	$< 0r0; 0/1/- >$	14	$0r0$	1	0	–	$< 0r0; 1/0/- >$
15	$1r1$	0	1	–	$< 1r1; 0/1/- >$	16	$1r1$	1	0	–	$< 1r1; 1/0/- >$
17	0	$0w0$	1	–	$< 0; 0w0/1/- >$	18	1	$0w0$	1	–	$< 1; 0w0/1/- >$
19	0	$0w1$	0	–	$< 0; 0w1/0/- >$	20	1	$0w1$	0	–	$< 1; 0w1/0/- >$
21	0	$1w0$	1	–	$< 0; 1w0/1/- >$	22	1	$1w0$	1	–	$< 1; 1w0/1/- >$
23	0	$1w1$	0	–	$< 0; 1w1/0/- >$	24	1	$1w1$	0	–	$< 1; 1w1/0/- >$
25	0	$0r0$	0	1	$< 0; 0r0/0/1 >$	26	1	$0r0$	0	1	$< 1; 0r0/0/1 >$
27	0	$0r0$	1	0	$< 0; 0r0/1/0 >$	28	1	$0r0$	1	0	$< 1; 0r0/1/0 >$
29	0	$0r0$	1	1	$< 0; 0r0/1/1 >$	30	1	$0r0$	1	1	$< 1; 0r0/1/1 >$
31	0	$1r1$	0	0	$< 0; 1r1/0/0 >$	32	1	$1r1$	0	0	$< 1; 1r1/0/0 >$
33	0	$1r1$	0	1	$< 0; 1r1/0/1 >$	34	1	$1r1$	0	1	$< 1; 1r1/0/1 >$
35	0	$1r1$	1	0	$< 0; 1r1/1/0 >$	36	1	$1r1$	1	0	$< 1; 1r1/1/0 >$

Applying the special notation of S, a two-cell static FP can be represented as follows $<S/F/R> = <S_a; S_v/F/R>_{a,v}$. Table 3 enumerates the 36 possible two-cell static FPs this notation can distinguish.

Below, a list of FFMs, some well known and some new, is constructed from the 36 FPs of Table 3. The new FFMs below are defined in such a way that all FPs are covered by at least one FFM.

1. **State coupling fault (CFst)**—Two cells are said to have a CFst if the victim is forced into a given logic state only if the aggressor is in a given state, without performing any operation on the victim. 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. Four types of CFst exist which can be

summed up as: $\text{CFst}_{x;y} = \{<x;y/\overline{y}/->\}$, where x, $y \in \{0,1\}$.

2. **Idempotent coupling fault (CFid)**—Two cells are said to have an CFid if a transition write operation ($0w1$ and $1w0$) on the aggressor forces the victim into a given state. This fault is sensitized by a transition write operation performed on the aggressor. Four types of CFid exist which can be summed up as: $\text{CFid}_{xw\overline{x};y} = \{<xw\overline{x};y/\overline{y}/->\}$, where x, $y \in \{0,1\}$.
3. **Inversion coupling fault (CFin)**—Two cells are said to have an CFin if the logic value of the victim is inverted in case a transition write operation is performed on the aggressor. Two types of CFin exist which can be summed up as: $\text{CFin}_{xw\overline{x}} = \{<xw\overline{x};y/\overline{y}/->, <xw\overline{x};\overline{y}/y/->\}$, where x, $y \in \{0,1\}$.
4. **Non-transition coupling fault (CFnt)**—Two cells are said to suffer from a CFnt if a non-transition write operation ($0w0$ and $1w1$) performed on the aggressor forces the victim into a given state. Four types of CFnt exist which can summed up as: $\text{CFnt}_{xwx;y} = \{<xwx;y/\overline{y}/->\}$, where x, $y \in \{0,1\}$.
5. **Disturb coupling fault (CFds)**—Two cells are said to have a CFds if an operation (write or read) performed on the aggressor forces the victim into a given logic state. Here, any operation performed on the aggressor is accepted as a sensitizing operation for the fault, be it a read, a transition write or a non-transition write operation. Twelve types of CFds exist which can be summed up as: $\text{CFds}_{xwy;z} = \{<xwy;z/\overline{z}/->\}$ and $\text{CFds}_{xrx;y} = \{<xrx;y/\overline{y}/->\}$, where x, y, $z \in \{0,1\}$.
6. **Transition coupling fault (CFtr)**—Two cells are said to have a CFtr if a given logic value in the aggressor results in the failure of a transition write operation performed on the victim. This fault is sensitized by a write operation on the victim and setting the aggressor into a given state. Four types of CFtr exist which can be summed up as: $\text{CFtr}_{x;\uparrow} = \{<x;0w1/0/->\}$ and $\text{CFtr}_{x;\downarrow} = \{<x;1w0/1/->\}$, where $x \in \{0,1\}$.
7. **Write disturb coupling fault (CFwd)**—A cell is said to have a CFwd if a non-transition write operation performed on the victim results in a transition when the aggressor is set into a given logic state. Four types of CFwd exist: $\text{CFwd}_{x;y} = \{<x;ywy/\overline{y}/->\}$, where x, $y \in \{0,1\}$.
8. **Read disturb coupling fault (CFrd)**—Two cells are said to have a CFrd if a read operation performed on the victim destroys the data stored in the victim if a given state is present in the aggressor. Four types of CFrd exist: $\text{CFrd}_{x;y} = \{<x;yry/\overline{y}/\overline{y}>\}$, where x, $y \in \{0,1\}$.
9. **Incorrect read coupling fault (CFir)**—Two cells are said to have an CFir if a read operation performed on the victim returns the incorrect logic value when the aggressor is set into a given state. Four types of CFir exist: $\text{CFir}_{x;y} = \{<x;yry/y/\overline{y}>\}$, where x, $y \in \{0,1\}$.
10. **Deceptive read disturb coupling fault (CFdr)**—A cell is said to have a CFdr if a read operation performed on the victim returns the correct logic value and changes the contents of the victim, when the aggressor is set into a given logic state. Four types of CFdr exist: $\text{CFdr}_{x;y} = \{<x;yry/\overline{y}/y>\}$, where x, $y \in \{0,1\}$.

There is a need to select a collection of the FFMs defined above that would cover all FPs listed in Table 3. An analysis of the defined FFMs shows that the FFMs CFst, CFds, CFtr, CFrd, CFir, CFdr and CFwd are necessary and sufficient to cover all two-cell static FPs. Moreover, no other combination of FFMs may be constructed with this property. Any two-cell static FFM can be represented as the union of two or more of these 36 FFMs. For example, if a defect results in a faulty behavior represented by an incorrect read coupling fault $\{<1;0r0/0/1>\}$ and a read disturb coupling fault $\{<1;1r1/0/0>\}$, then the corresponding behavior is presented as: $\{<1;0r0/0/1>\} \cup \{<1;1r1/0/0>\} = \{<1;0r0/0/1>, <1;1r1/0/0>\}$.

5.2 Dynamic two-cell FFMs

Just like the case of single-cell dynamic FFMs, we restrict ourselves here to the analysis of 2-operation dynamic fault models. Any particular FP is denoted by $<S/F/R>$, where for two-cell 2-operation dynamic FPs, S can be one of the following:

$$\begin{aligned} S_{aa} &= a(iO_1d_1O_2d_2)\ v(j) \\ S_{av} &= a(iO_1d_1)\ v(jO_2d_2) \\ S_{va} &= v(jO_1d_1)\ a(iO_2d_2) \\ S_{vv} &= a(i)\ v(jO_1d_1O_2d_2) \end{aligned}$$

The subscripts a and v in the SOS names indicate whether each of the two operations is performed on the aggressor or the victim, respectively. For example, $<v(0r0)\ a(1r1)/1/->$ stands for an FP sensitized by performing a $0r0$ first on the victim then performing a $1r1$ on the aggressor. After performing the sensitizing sequence, a 1 is detected in the victim cell instead of the expected 0.

Based on the values of S, F and R, all 192 detectable two-cell 2-operation dynamic FPs are compiled in Table 4. The listed FPs can be generated automatically by substituting all possible values of S, F and R into the FP notation and discarding those that do not represent a faulty behavior ($<a(0w0w0)v(0)/0/->$, for example).

Below, the FPs are used to define two-cell 2-operation FFMs. Each FP can be used to define a corresponding FFM resulting in 192 dynamic FFMs. Since we will only attempt to verify a limited number of dynamic FFMs (see Section 6), only those FPs indicated by an asterisk (*) in Table 4 are used to define corresponding FFMs.

1. **Dynamic read disturb coupling faults (CFrd$_{x;yz}$)**—The dynamic CFrd has a definition similar to the CFrd in the static case. It is a fault whereby an $a(x)$ $v(ywyry)$ or $a(x)$ $v(\overline{y}wyry)$ SOS changes the stored logic value in the victim to $\overline{y}$ and gives an incorrect output. Eight types of dynamic CFrd exist which can be summed up as: CFrd$_{x;yz}$ = $\{<a(x)\ v(ywzrz)/\overline{z}/\overline{z}>\}$, where x, y, $z \in \{0, 1\}$.
2. **Dynamic incorrect read coupling faults (CFir$_{x;yz}$)**—The dynamic CFir has a definition similar to the CFir in the static case. It is a fault whereby an $a(x)$ $v(ywyry)$ or $a(x)$ $v(\overline{y}wyry)$ SOS returns the logic value $\overline{y}$ while keeping the correct state of the cell. Eight types of dynamic CFir exist which can be summed up as: CFir$_{x;yz}$ = $\{<a(x)\ v(ywzrz)/z/\overline{z}>\}$, where x, y, $z \in \{0, 1\}$.
3. **Dynamic deceptive read disturb coupling faults (CFdr$_{x;yz}$)**—The dynamic CFdr has a definition similar to the CFdr in the static case. It is a fault whereby an $a(x)$ $v(ywyry)$ or $a(x)$ $v(\overline{y}wyry)$ SOS returns the correct logic value y while destroying the state of the cell. Eight types of dynamic CFdr exist which can be summed up as: CFdr$_{x;yz}$ = $\{<a(x)\ v(ywzrz)/\overline{z}/z>\}$, where x, y, $z \in \{0, 1\}$.

6 Verification of FFMs

At this point, the static and dynamic FFMs defined in Sections 4 and 5 are to be verified. To this end, a study has been performed to analyze the static and dynamic faulty behavior of DRAM devices, based on defect injection and simulation [Al-Ars99]. The simulation tool used is Pstar which is an analogue simulator produced by Philips. The simulation model is a clone of a real embedded DRAM design-validation model. The model has been simplified to include only one folded cell array column (2×2 memory cells, 2 reference cells, precharge devices and a sense amplifier), one write driver and one data output buffer. In the following, the simulation methodology is discussed first, followed by the results of the simulation.

6.1 Simulation methodology

Here, we discuss first the types and specifications of the injected defects, then the FPs targeted by the simulation are identified.

Injected defects: The injected defects analyzed are opens, shorts and bridges. Opens are resistive components with any resistance ($0\ \Omega < R_{op} < \infty\ \Omega$), injected at all possible locations within memory cells, along bit lines, on word lines, and in the sense amplifier. Shorts are resistive or capacitive components with certain values ($0\ \Omega < R_{sh} < \infty\ \Omega$, $0\ \text{F} < C_{sh} < C_{max}$), injected at all possible locations within memory cells and along bit lines. Bridges have the same specifications as shorts, but are only injected within and between memory cells. No layout information has been used since all possible opens, shorts and bridges are considered at the electrical level.

The analysis procedure for opens takes all open resistances and initial floating node voltages into consideration. Inserting an open with a high resistance can create floating nodes from the nodes connected to the open. The analysis varies the voltage on these nodes between the upper and lower voltage limits. For shorts and bridges, the analysis procedure takes all possible resistances and capacitances into consideration.

Targeted FPs: All static FPs described in Tables 1 and 3 are taken into consideration. On the other hand, only a number of dynamic FPs are considered. It is clear from Tables 2 and 4 that, although we restricted our analysis to the single-cell and two-cell 2-operation dynamic behavior, the number of related FPs is still relatively high. We choose to limit our attention to the single-cell dynamic SOS's $xwxrx$ and $\overline{x}wxrx$, where $x \in \{0, 1\}$. The chosen four SOS's are capable of sensitizing 12 single-cell 2-operation dynamic FPs. In the same way, we choose to limit our attention to the two-cell dynamic SOS's $a(x)$ $v(ywyry)$ and $a(x)$ $v(\overline{y}wyry)$, where x and $y \in \{0, 1\}$. The chosen 8 two-cell dy-

Table 4. All possible combinations of the values in the $<S/F/R>$ notation resulting in a two-cell 2-operation dynamic FP.

< S/F/R >	< S/F/R >	< S/F/R >	< S/F/R >	< S/F/R >
< a(0w0w0) v(0)/1/− >	< a(0w0w0) v(1)/0/− >	< a(0w0w1) v(0)/1/− >	< a(0w0w1) v(1)/0/− >	< a(0w0r0) v(0)/1/− >
< a(0w0r0) v(1)/0/− >	< a(0w1w0) v(0)/1/− >	< a(0w1w0) v(1)/0/− >	< a(0w1w1) v(0)/1/− >	< a(0w1w1) v(1)/0/− >
< a(0w1r1) v(0)/1/− >	< a(0w1r1) v(1)/0/− >	< a(1w0w0) v(0)/1/− >	< a(1w0w0) v(1)/0/− >	< a(1w0w1) v(0)/1/− >
< a(1w0w1) v(1)/0/− >	< a(1w0r0) v(0)/1/− >	< a(1w0r0) v(1)/0/− >	< a(1w1w0) v(0)/1/− >	< a(1w1w0) v(1)/0/− >
< a(1w1w1) v(0)/1/− >	< a(1w1w1) v(1)/0/− >	< a(1w1r1) v(0)/1/− >	< a(1w1r1) v(1)/0/− >	< a(0r0w0) v(0)/1/− >
< a(0r0w0) v(1)/0/− >	< a(0r0w1) v(0)/1/− >	< a(0r0w1) v(1)/0/− >	< a(0r0r0) v(0)/1/− >	< a(0r0r0) v(1)/1/− >
< a(1r1w0) v(0)/1/− >	< a(1r1w0) v(1)/0/− >	< a(1r1w1) v(0)/1/− >	< a(1r1w1) v(1)/0/− >	< a(1r1r1) v(0)/1/− >
< a(1r1r1) v(1)/0/− >				
< a(0) v(0w0w0)/1/− >*	< a(1) v(0w0w0)/1/− >*	< a(0) v(0w0w1)/0/− >*	< a(1) v(0w0w1)/0/− >*	< a(0) v(0w0r0)/0/1 >*
< a(1) v(0w0r0)/0/1 >*	< a(0) v(0w0r0)/1/0 >*	< a(1) v(0w0r0)/1/0 >*	< a(0) v(0w0r0)/1/1 >*	< a(1) v(0w0r0)/1/1 >*
< a(0) v(0w1w0)/1/− >*	< a(1) v(0w1w0)/1/− >*	< a(0) v(0w1w1)/0/− >*	< a(1) v(0w1w1)/0/− >*	< a(0) v(0w1r1)/0/0 >*
< a(1) v(0w1r1)/0/0 >*	< a(0) v(0w1r1)/0/1 >*	< a(1) v(0w1r1)/0/1 >*	< a(0) v(0w1r1)/1/0 >*	< a(1) v(0w1r1)/1/0 >*
< a(0) v(1w0w0)/1/− >*	< a(1) v(1w0w0)/1/− >*	< a(0) v(1w0w1)/0/− >*	< a(1) v(1w0w1)/0/− >*	< a(0) v(1w0r0)/0/1 >*
< a(1) v(1w0r0)/0/1 >*	< a(0) v(1w0r0)/1/0 >*	< a(1) v(1w0r0)/1/0 >*	< a(0) v(1w0r0)/1/1 >*	< a(1) v(1w0r0)/1/1 >*
< a(0) v(1w1w0)/1/− >*	< a(1) v(1w1w0)/1/− >*	< a(0) v(1w1w1)/0/− >*	< a(1) v(1w1w1)/0/− >*	< a(0) v(1w1r1)/0/0 >*
< a(1) v(1w1r1)/0/0 >*	< a(0) v(1w1r1)/0/1 >*	< a(1) v(1w1r1)/0/1 >*	< a(0) v(1w1r1)/1/0 >*	< a(1) v(1w1r1)/1/0 >*
< a(0) v(0r0w0)/1/− >*	< a(1) v(0r0w0)/1/− >*	< a(0) v(0r0w1)/0/− >*	< a(1) v(0r0w1)/0/− >*	< a(0) v(0r0r0)/0/1 >*
< a(1) v(0r0r0)/0/1 >*	< a(0) v(0r0r0)/1/0 >*	< a(1) v(0r0r0)/1/0 >*	< a(0) v(0r0r0)/1/1 >*	< a(1) v(0r0r0)/1/1 >*
< a(0) v(1r1w0)/1/− >*	< a(1) v(1r1w0)/1/− >*	< a(0) v(1r1w1)/0/− >*	< a(1) v(1r1w1)/0/− >*	< a(0) v(1r1r1)/0/0 >*
< a(1) v(1r1r1)/0/0 >*	< a(0) v(1r1r1)/0/1 >*	< a(1) v(1r1r1)/0/1 >*	< a(0) v(1r1r1)/1/1 >*	< a(1) v(1r1r1)/1/1 >*
< a(0w0) v(0w0)/1/− >	< a(1w0) v(0w0)/1/− >	< a(0w0) v(0w1)/0/− >	< a(1w0) v(0w1)/0/− >	< a(0w0) v(0r0)/0/1 >
< a(1w0) v(0r0)/0/1 >	< a(0w0) v(0r0)/1/0 >	< a(1w0) v(0r0)/1/0 >	< a(0w0) v(0r0)/1/1 >	< a(1w0) v(0r0)/1/1 >
< a(0w0) v(1w0)/1/− >	< a(1w0) v(1w0)/1/− >	< a(0w0) v(1w1)/0/− >	< a(1w0) v(1w1)/0/− >	< a(0w0) v(1r1)/0/0 >
< a(1w0) v(1r1)/0/0 >	< a(0w0) v(1r1)/0/1 >	< a(1w0) v(1r1)/0/1 >	< a(0w0) v(1r1)/1/0 >	< a(1w0) v(1r1)/1/0 >
< a(0w1) v(0w0)/1/− >	< a(1w1) v(0w0)/1/− >	< a(0w1) v(0w1)/0/− >	< a(1w1) v(0w1)/0/− >	< a(0w1) v(0r0)/0/1 >
< a(1w1) v(0r0)/0/1 >	< a(0w1) v(0r0)/1/0 >	< a(1w1) v(0r0)/1/0 >	< a(0w1) v(0r0)/1/1 >	< a(1w1) v(0r0)/1/1 >
< a(0w1) v(1w0)/1/− >	< a(1w1) v(1w0)/1/− >	< a(0w1) v(1w1)/0/− >	< a(1w1) v(1w1)/0/− >	< a(0w1) v(1r1)/0/0 >
< a(1w1) v(1r1)/0/0 >	< a(0w1) v(1r1)/0/1 >	< a(1w1) v(1r1)/0/1 >	< a(0w1) v(1r1)/1/0 >	< a(1w1) v(1r1)/1/0 >
< a(0r0) v(0w0)/1/− >	< a(1r1) v(0w0)/1/− >	< a(0r0) v(0w1)/0/− >	< a(1r1) v(0w1)/0/− >	< a(0r0) v(0r0)/0/1 >
< a(1r1) v(0r0)/0/1 >	< a(0r0) v(0r0)/1/0 >	< a(1r1) v(0r0)/1/0 >	< a(0r0) v(0r0)/1/1 >	< a(1r1) v(0r0)/1/1 >
< a(0r0) v(1w0)/1/− >	< a(1r1) v(1w0)/1/− >	< a(0r0) v(1w1)/0/− >	< a(1r1) v(1w1)/0/− >	< a(0r0) v(1r1)/0/0 >
< a(1r1) v(1r1)/0/0 >	< a(0r0) v(1r1)/0/1 >	< a(1r1) v(1r1)/0/1 >	< a(0r0) v(1r1)/1/1 >	< a(1r1) v(1r1)/1/1 >
< v(0w0) a(0w0)/1/− >	< v(1w0) a(0w0)/1/− >	< v(0w0) a(0w1)/1/− >	< v(1w0) a(0w1)/1/− >	< v(0w0) a(0r0)/1/− >
< v(1w0) a(0r0)/1/− >	< v(0w0) a(1w0)/1/− >	< v(1w0) a(1w0)/1/− >	< v(0w0) a(1w1)/1/− >	< v(1w0) a(1w1)/1/− >
< v(0w0) a(1r1)/1/− >	< v(1w0) a(1r1)/1/− >	< v(0w1) a(0w0)/0/− >	< v(1w1) a(0w0)/0/− >	< v(0w1) a(0w1)/0/− >
< v(1w1) a(0w1)/0/− >	< v(0w1) a(0r0)/0/− >	< v(1w1) a(0r0)/0/− >	< v(0w1) a(1w0)/0/− >	< v(1w1) a(1w0)/0/− >
< v(0w1) a(1w1)/0/− >	< v(1w1) a(1w1)/0/− >	< v(0w1) a(1r1)/0/− >	< v(1w1) a(1r1)/0/− >	< v(0r0) a(0w0)/1/− >
< v(1r1) a(0w0)/0/− >	< v(0r0) a(0w1)/1/− >	< v(1r1) a(0w1)/0/− >	< v(0r0) a(0r0)/1/− >	< v(1r1) a(0r0)/0/− >
< v(0r0) a(1w0)/1/− >	< v(1r1) a(1w0)/0/− >	< v(0r0) a(1w1)/1/− >	< v(1r1) a(1w1)/0/− >	< v(0r0) a(1r1)/0/0 >
< v(1r1) a(1r1)/0/− >				

namic SOS's are capable of sensitizing 24 2-operation dynamic FPs. The reason for selecting these sequences in particular is the fact that, in memory devices, an isolated write operation is practically not enough to detect a fault since, externally, a cell needs to be read to detect the stored value set during the write.

6.2 Simulation results

The performed simulations on the DRAM model result in many types of single-cell, two-cell, static and dynamic faulty behavior. The results of the analysis are discussed in this section.

Detected FPs: The fault analysis shows that all static single-cell FPs listed in Table 1 have been detected. Moreover, all *targeted* 2-operation single-cell FPs #3–#5, #8–#10, #13–#15 and #18–#20 of Table 2 have been detected. Out of the static two-cell FPs listed in Table 3, FPs #2–#16 and #30–#32 have been detected. Finally, none of the targeted 2-operation two-cell FPs (indicated by an asterisk in Table 4) have been detected.

Discussion: Table 5 lists the detected single-cell FFMs as a result of defect injection and simulation. The second column states the defects resulting in the FFMs, where C stands for memory cells, BL stands for bit lines, WL stands for word lines, SA stands for sense amplifier, WC stands for within cells and BC stands for between cells. The third column states the number of times the FFMs are detected, such that every time an individual FFM is detected the count is increased by one (for example, the count of SF$_x$ is increased by one if SF$_1$ or SF$_0$ is detected due to a given defect). The table shows that the read disturb fault has the highest count. This is because read operations depend on detecting small voltage differences on a bit line pair which is easily disturbed.

Table 6 lists the detected two-cell FFMs. Every time an individual FFM is detected, the count in the second column is increased by one (for example, CFst is increased by one if either CFst$_{0;0}$, CFst$_{0;1}$, CFst$_{1;0}$ or CFst$_{1;1}$ is detected due to a given defect). The table shows that the disturb coupling fault has the highest count, which is because CFds describes the highest number of individual FFMs among all other two-cell faults (12 FFMs). Not all static two-cell FPs defined in Table 3 have been detected. The undetected FPs are those covered by CFst$_{0;0}$, CFrd$_{0;0}$, all CFtr FFMs, all CFwd FFMs, all CFir FFMs and all CFdr FFMs. Dynamic two-cell FPs have not been detected either, which can be explained as follows. The SOS's associated with the targeted two-cell dynamic FPs consist

Table 5. Summary of the detected static and dynamic single-cell FFMs as a result of defect injection and simulation.

FFMs	Defect	Count
SF_x	Opens: WL	2
	Shorts: C	4
	Bridges: WC, BC	12
TF_x	Opens: C, BL, WL, SA	36
	Shorts: C, BL	8
	Bridges: WC, BC	12
WDF_x	Opens: C, BL, SA	24
	Shorts: C, BL	8
	Bridges: WC, BC	12
RDF_x	Opens: C, BL, WL, SA	41
	Shorts: C, BL	10
	Bridges: WC, BC	12
IRF_x	Opens: C, BL, WL, SA	20
	Shorts: —	0
	Bridges: WC	4
$DRDF_x$	Opens: BL, SA	14
	Shorts: —	0
	Bridges: WC	8
RDF_{xy}	Opens: C, BL, WL, SA	58
	Shorts: C, BL	20
	Bridges: WC, BC	20
IRF_{xy}	Opens: C, BL, WL	22
	Shorts: —	0
	Bridges: WC	8
$DRDF_{xy}$	Opens: C, BL, SA	38
	Shorts: —	0
	Bridges: WC	8

of a sequence of write then read operations performed on the victim. Since no write operation on the victim results in a coupling fault, it is not expected that a subsequent read operation would cause a coupling fault either.

7 Conclusions

In this paper, the notion of fault primitives (FPs), together with their notation, has been introduced. The traditional functional fault models have been shown to be composed of FPs. A taxonomy, based on the proposed FPs, has been introduced for all possible memory faults FPs can describe. This taxonomy shows that the space of possible memory faults is infinite, and that it contains the currently established functional faults. FPs are perfectly suited for test purposes, since they precisely define the operation sequences required to detect the faults the FPs describe.

Resistive and capacitive defect injection and simulation of a DRAM model showed the presence of established FFMs: TF_x, RDF_x, $DRDF_x$, CFst, CFds, CFnt, CFid, and CFrd. The simulation also identified a number of the newly defined FFMs: SF_x, IRF_x, RDF_{xy}, IRF_{xy} and $DRDF_{xy}$.

Table 6. Summary of the detected static two-cell FFMs as a result of defect injection and simulation.

FFMs	Count
CFst	12
CFds	84
CFnt	24
CFid	36
CFrd	12

References

[Adams96] R.D. Adams and E.S. Cooley, "Analysis of a Deceptive Destructive Read Memory Fault Model and Recommended Testing," *In Proc. IEEE North Atlantic Test Workshop*, 1996.

[Al-Ars99] Z. Al-Ars, *Analysis of The Space of Functional Fault Models and Its Application to Embedded DRAMs*, Technical Report No. 1-68340-28(1999)-07, CARDIT, Delft University of Technology, Delft, The Netherlands, 1999.

[Dekker90] R. Dekker *et al.*, "A Realistic Fault Model and Test Algorithms for Static Random Access Memories," *In IEEE Trans. on Computers*, C-9(6), 1990, pp. 567–572.

[Mak98] T.M. Mak, "Cache RAM Inductive Fault Analysis with Fab Defect Modeling," *In Proc. IEEE International Test Conference*, 1998, pp. 862–871.

[Schanstra99] Ivo Schanstra and Ad J. van de Goor, "Industrial Evaluation of Stress Combinations for March Tests applied to SRAMs," *In Proc. IEEE International Test Conference*, 1999, pp. 983–992.

[vdGoor94] A.J. van de Goor and B. Smit, "Generating Memory Tests Automatically," *In Proc. IEEE International Test Conference*, 1994, pp. 870–878.

[vdGoor96] A. 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.

[vdGoor98] A.J. van de Goor, *Testing Semiconductor Memories, Theory and Practice*, ComTex Publishing, Gouda, The Netherlands, 1998, e-mail: vdGoor@cardit.et.tudelft.nl

[vdGoor99a] Ad J. van de Goor and J. de Neef, "Industrial Evaluation of DRAM Tests," *In Proc. Design, Automation and Test in Europe*, 1999, pp. 623–630.

[vdGoor99b] A.J. van de Goor and J.E. Simonse, "Defining SRAM Resistive Defects and Their Simulation Stimuli," *In Proc. Asian Test Symposium*, 1999, pp. 33–40.

[Zarrineh98] K. Zarrineh *et al.*, "A New Framework for Generating Optimal March Tests for Memory Arrays," *In Proc. IEEE International Test Conference*, 1998, pp. 73–82.