Test et Diagnostic de Fautes Dynamiques dans les Mémoires SRAM

Alexandre Ney

UNIVERSITE MONTPELLIER II
SCIENCES ET TECHNIQUES DU LANGUEDOC

T H E S E

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE MONTPELLIER II

Discipline : Microélectronique

Ecole Doctorale : Information, Structure, Systèmes

présentée et soutenue publiquement

par

Alexandre NEY

Le 29 Septembre 2008

Titre :

Test et Diagnostic de Fautes Dynamiques dans les Mémoires SRAM

JURY

Serge Pravossoudovitch	Président
Patrick Girard	Directeur de Thèse
Dominique Dallet	Rapporteur
Christophe Muller	Rapporteur
Jean-Christophe Vial	Examinateur
Arnaud Virazel	Examinateur

À mes parents et mon amie dont le soutien et la présence m'ont permis de mener à bien cette thèse

Remerciements

Cette thèse a été effectuée au laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier (LIRMM), dirigée par Monsieur Michel Robert, Professeur à l'Université de Montpellier II, dans le département de Microélectronique dont le responsable est Monsieur Lionel Torres, Professeur à l'Université de Montpellier II. Je les remercie de m'avoir accueilli.

Je souhaite exprimer ma sympathie à l'égard de Patrick Girard, Directeur de Recherche au CNRS, mon directeur de thèse, dont l'encadrement et les encouragements m'ont permis de mener à bien ce projet de recherche. Un grand merci à Serge Pravossoudovitch, Professeur à l'Université de Montpellier II, mon co-directeur de thèse

Je remercie également Arnaud Virazel et Alberto Bosio pour leur soutien aussi bien technique qu'humain.

Je tenais aussi à remercier Jean-Christophe Vial, directeur de l'équipe mémoire embarquée de la société Infineon, de m'avoir accueilli au sein de son équipe tout au long de ma thèse. Je n'oublie pas d'exprimer ma gratitude à Magali Bastian, Vincent Gouin et Christophe Chanussot, ingénieurs à Infineon.

Merci également à Dominique Dallet, Professeur à l'ENSEIRB de Bordeaux, et Christophe Muller, Professeur au L2MP de Marseille, de s'être intéressés à ce travail et d'avoir accepté d'en être les rapporteurs.

Enfin, je terminerais en remerciant Julien Vial (dit le King ou p'tit bidou n°2), Boris Alandry (dit Didier), Nico Saint-Jean (dit p'tit bidou n°1), Lionel Gouyet (dit Shadow), Mehdi (doigts), Nico Houarche (dit Bernardo), Eric Ze Qin, Victor Lomne, Yohan Guillemenet, Amine Debhaoui, Monsieur Ginez, Alexandre Rousset, Marion Doulcier, Olivier Leman, Youssef Benabboud (dit Zizou), Jean-Etienne Lorival, Nico Bruchon, Nico Pous, Alin Razafindraeibé, Jean-Baptiste Lerat, …

Index

Functional Fault model Glossary

Address Decoder Open Fault (ADOF): *A decoder is said to have an ADOF when changing only one bit on its address results in selecting this new address but also the previous one. Consequently, two core-cells are selected at the same time for a read or a write operation.*

dynamic Read Destructive Fault (dRDF): *A core-cell is said to have a dRDF if a write operation immediately followed by a read operation performed on the core-cell changes the logic state of this core-cell and returns and incorrect value on the output.*

dynamic two-cells Incorrect Read Fault type 1 (d2cIRF1): *A sense amplifier is said to have a d2cIRF1 if it is unable to read any value. So, the read data value at the output is the one previously stored in the data output circuitry. This is a two-cell fault model as it requires two read operations on two distinct core-cells.*

dynamic two-cells Incorrect Read Fault type 2 (d2cIRF2): *A sense amplifier is said to have a d2cIRF2 if it is only able to perform a* $r0$ *or* $r1$ *operation As for d2cIRF1, this is a two-cell fault model as it requires two read operations on two distinct core-cells.*

Incorrect Read Fault (IRF): *A core-cell is said to have an IRF if a read operation performed on the cell returns an incorrect logic value, while keeping the correct stored value in the cell.*

Read Destructive Fault (RDF): *A core-cell is said to have a RDF if a read operation performed on the cell changes the data in the cell and returns an incorrect value on the output.*

Slow Write Driver Fault (SWDF): *A write driver is said to have a SWDF if it cannot act a w0 (w1) when this operation is preceded by a w1 (w0). That results on the core-cell that does not change its data content.*

Stuck-At Fault (SAF): *A core-cell is said to have a SAF if its content is always at a given value and cannot be changed to the opposite state*

Transition Fault (TF): *A core-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.*

Un-Restored Write Fault (URWF): *The pull up of one of the two bit lines is not completely achieved after the state reached with a write operation. Consequently the following read operation of an opposite data in a cell belonging to the same I/O circuitry is not correctly acted.*

Un-Restored Destructive Write Fault (URDWF): *The same definition as URWF but in addition to the faulty read operation, the core-cell flips.*

Short Acronym Dictionary

ADOF:	**Address Decoder Open Fault**
BL:	**Bit Line**
CF:	**Coupling Fault**
d2cIRF:	**dynamic two-cell Incorrect Read Fault**
Df:	**Defect**
DFD:	**Design For Diagnosis**
DOF:	**Degree Of Freedom**
DR:	**Diagnosability Ratio**
dRDF:	**dynamic Read Destructive Fault**
FFM:	**Functional Fault Model**
FP:	**Fault Primitive**
MT:	**March Test**
RDF:	**Read Destructive Fault**
SAF:	**Stuck-At Fault**
SoC:	**System on Chip**
SOS:	**Sensitizing Operation Sequence**
SRAM:	**Static Random Access Memory**
SWDF:	**Slow Write Driver Fault**
TF:	**Transition Fault**
URDWF:	**Un-Restored Destructive Write Fault**
URWF:	**Un-Restored Write Fault**
VDSM:	**Very Deep Sub-Micron**
WL:	**Word Line**

General Introduction

In the actual landscape of System-on-Chips (SoC), there is a wide panel of available memories due to the high demand of storage in different kind of applications and systems. These semiconductor memories are classified in two families: the volatile memories and the non-volatile ones as shown in Figure 1.

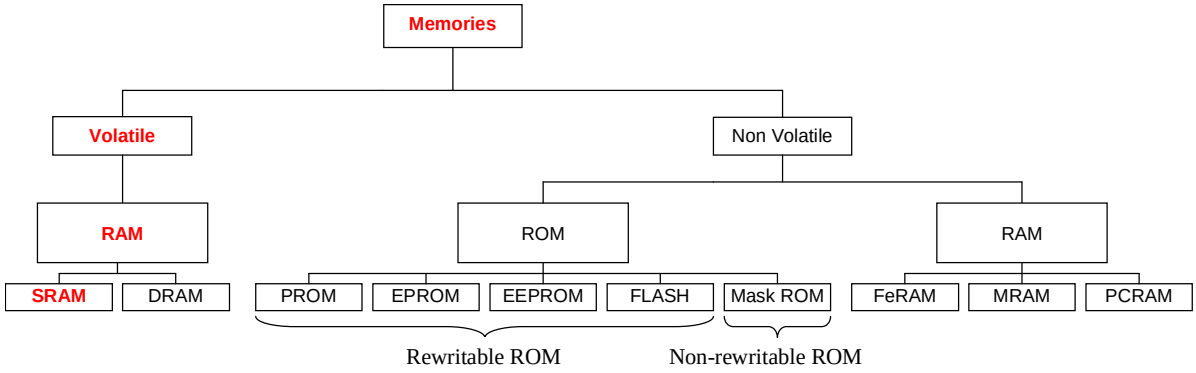

Figure 1 – Memory classification

Volatile memories have the particularity to lose their content when the power supply is turned off. Those are based on Random Access Memories (RAM) concept meaning an arbitrary access. There are two kinds of volatile RAMs, the Static RAMs (SRAMs) and the Dynamic RAMs (DRAMs). SRAMs keep automatically their contents while power is turned on and present a very short access time compare to DRAMs. Consequently, they are used for fast applications such as cache memories for processor. On the other hand, DRAM contents need to be refreshed periodically. Nevertheless, their high integration density compare to SRAMs makes DRAMs more useful for mass data storage.

By opposition to volatile memories, non-volatile ones keep their data indefinitely (theoretically), even if the power is turned off. These kinds of memories are divided in two sub-families, the one based on the ROM (Read Only Memories) concept, the second based on the RAM concept (previously presented). Originally, ROMs were the only non-volatile memories and their contents were not rewritable. However, researchers have found new mechanisms and materials allowing these memories to be rewritable (PROM, EPROM…). Recently, non-volatile memories based on the RAM principle have been developed. They present the same organization as volatile RAMs, except that they use materials allowing to keep data even if the power is turned off. For example, in MRAM, magnetic materials are used to store data as magnetic field. The polarity of this magnetic field determines the stored logic data (‘0’ or ‘1’).

Among existing memory types, this thesis is dedicated to SRAMs testing as they are widely used in embedded and high speed applications. In SoCs, both the number of embedded memory cores and area occupied are rapidly increasing. According to the SIA roadmap [SIA05], memories should occupy 94% of SoC silicon area in the next ten years (see Figure 2) making them the main detractor of SoC yield. In addition, SRAM core-cells are often designed by violating some layout rules to save area. They are also considered as a vehicle for CMOS process technology development. Advances in their fabrication, through the scaling for higher densities and faster speeds, is helpful for the performance establishment of other digital circuits. Considering this context, faults are more likely to happen in memories than in any other SoC part. Hence, efficient test and diagnosis methods for embedded SRAMs are therefore needed to reach a satisfactory SoC yield.

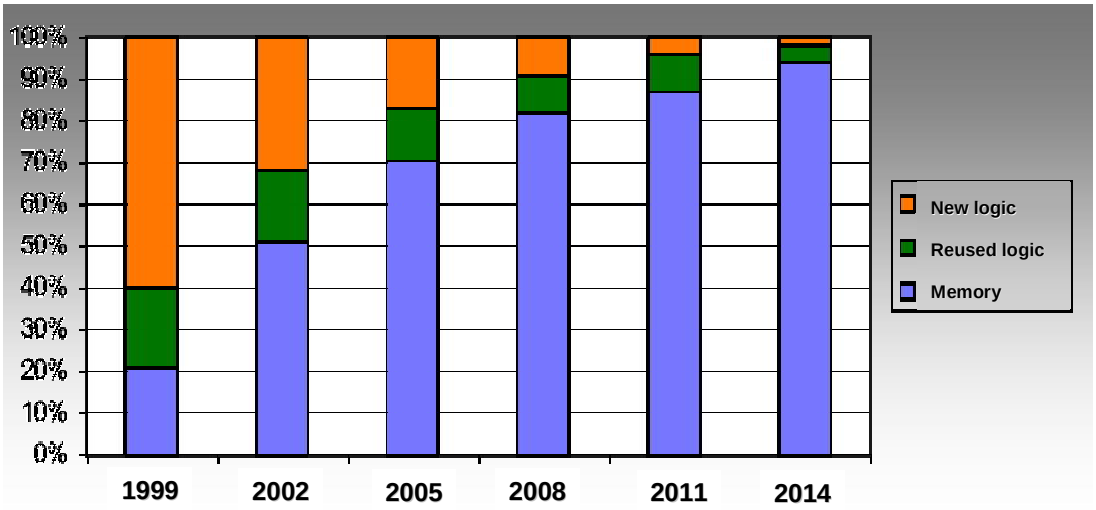

Figure 2 – ITRS roadmap: International Technology Roadmap for Semiconductors

The first part of this thesis is dedicated to SRAM testing solutions. Current test methods used for SRAMs are generally based on static fault detection such as stuck-at faults (SAF), transition faults (TF) and coupling faults (CF) [VDG98]. Static faults require at most one read/write operation to be sensitized. Specific tests, called March tests, are constructed to detect such fault types [VDG98]. However, in Very Deep SubMicron (VDSM) technologies, a new type of faulty behavior, called **dynamic faults** [VDG00, ARS01], are more likely to occur. These faults require more than one read/write operations in sequence to be sensitized and are most of the time undetectable with existing March tests. It has been shown that resistive-open defects (due to bad vias or contacts) are the main root cause of such faults. Resistive-open defect occurrences in the address decoder [DIL04a], core-cell [DIL04b, DIL05a] and pre-charge circuit [DIL05b, DIL06a] have already been analyzed.

A first objective of this thesis is to complete the previous studies by developing new and efficient solutions for dynamic fault induced by resistive-open defects in the write driver and in the sense amplifier of SRAMs.

The second part of this manuscript is oriented toward SRAM diagnosis solutions. In fact, as soon as the test phase has revealed logic errors in a given memory, diagnosis can be performed in order to precisely localize faulty sites. This may be helpful to improve memory yield by using redundancies [KIM98, HAR01, ZOR02] added to the SRAM structure. The repair phase consists in replacing defective blocks with spare ones. Some of the different types of redundancies include word redundancy, word line redundancy, bit line redundancy and I/O redundancy [RON02].

The diagnostic may also be used to improve yield ramp up for new technologies and new designs. This time, the crucial information is not only the fault localization but also fault type and its physical origin. With such information available, engineers can adjust the manufacturing process and/or enhance the memory design.

Existing diagnosis solutions are most of the time not able to precisely localize faulty sites, and to deal with dynamic faults.

The second objective of this thesis deals with new and efficient solutions for SRAMs diagnostic.

Works realized during the three years of this thesis have been carried out in collaboration with Infineon Technologies (Sophia Antipolis), under the framework of the NANOTEST – 2A702 European Project. With this partnership, we have been able to act a complete characterization of behaviors occurring in SRAMs. On the basis of this analysis and characterization work, we have determined various faults models and developed some efficient test solutions. Moreover, the industrial collaboration has also allowed the validation of all results. This work has been the object of several publications in international conferences and journals.

This manuscript is divided in two parts:

The first part is dedicated to dynamic faults testing in an SRAM, especially dynamic faults due to resistive-open defects in SRAM write drivers and sense amplifiers. Thereafter, we also show that process variations (called mismatches in case of local variations) on the transistor threshold voltage V_{TH} may impact the core-cell behavior.

The second part provides a presentation of new diagnosis techniques. In a first time, two Design For Diagnosis (DFD) methods able to deal with weak write drivers are presented. Next, a global memory diagnosis method, based on the use of algorithms, is proposed.

Part I: Test of dynamic faults in SRAMs

Introduction

Existing test solutions, based on March type algorithms, target functional fault models such as SAF, TF and CF faults models. Those are known as static fault models as they require at most one read/write operation to be sensitized. However, in VDSM technologies, dynamic faults [VDG00, ARS01] are more likely to occur. These faults require a specific read/write sequence to be sensitized and are mainly due to bad vias or contacts inducing resistive-open defects. Unfortunately, the existing March test solutions are most of the time unable to test these new kinds of faults [HAM03].

Some papers dealing with dynamic faults due to resistive-open defects in various blocks of the memory, such as address decoder [SAC97, DIL04a, DIL06b], core-cell [BOR03b, DIL04b, DIL05c, BOR05] and pre-charge circuit [ADA02, DIL05b, DIL07] have been proposed so far. However, there is a lack of studies on dynamic faults due to SRAM write driver and sense amplifier. We propose here to overcome that by proposing two studies on dynamic faults induced by resistive-open defects in the write driver and in the sense amplifier.

This part is organized as follows: a first chapter is dedicated to an overview of memory test. In the second chapter, the SRAM functioning is studied when write drivers are affected by resistive-open defects. The third chapter deals with memory functioning in presence of such defects in the sense amplifiers. Then, we will see in the fourth chapter that dynamic faults in SRAMs can also be due to local process variations also called mismatches. Finally, concluding remarks are provided in the last section.

Chapter 1. Background and state-of-the-art

This chapter gives the SRAM background useful for a complete understanding of the remaining of this part. Especially, a global view of a SRAM as well as the core-cell view are briefly depicted. Next, the techniques commonly used to test SRAMs are presented. Finally, a state-of-the-art on dynamic faults testing is provided in order to justify our study.

The organization of this chapter is as follows: in the first Section, a background on memory testing is provided. The second Section concerns the dynamic fault testing. Finally, Section 3 gives some conclusions.

I.1.1. Background on memory testing

I.1.1.1. SRAM structure

In any kind of memory, bits are either individually addressable (bit-oriented memories), or addressable by groups of 4, 8, 16 or more (word-oriented memories). For simplicity, we assume in our discussion that all the memory bits are individually addressable. The bulk of the memory consists of the cells in which the bits are stored. Each memory cell is an electronic circuit capable to store (at least) one bit.

The physical organization of the storage cells is commonly done in a square or nearly square matrix. In Figure I.1, we illustrate such organization. The cell matrix has 2^M rows and 2^N columns, for a total storage capacity of 2^{M+N} bits. For example, one Mega bits square matrix would have 1024 rows and 1024 columns (M = N = 10). Each core-cell in the array is connected to one of the 2^M row lines, universally called word lines (WL), and one of the 2^N column line, commonly called bit lines (BL). A particular core-cell can be accessed for a read or write operation by selecting its word line and its bit line.

The activation of one of the 2^M word lines is performed by the row decoder, which is a combinational logic circuit that selects the word line, corresponding to the address (in M bits) applied to its input. When the k^{th} word line is activated, whatever the operation, all the 2^N core-cells in the k^{th} row are connected to their respective bit lines. The connections of the couple of selected bit lines to the I/O circuitry is done by the column address decoder.

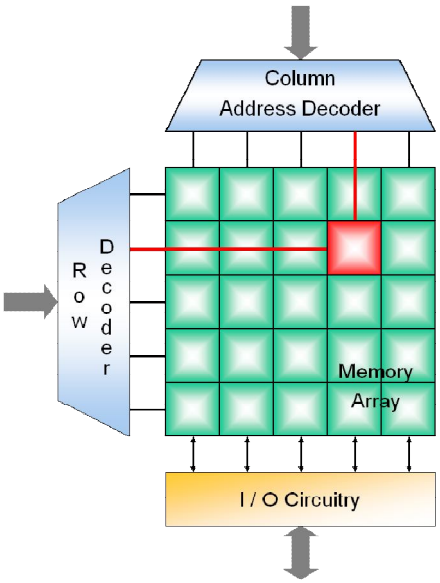

Figure I.1 – Scheme of the memory structure

The I/O circuitry is composed by a write driver and a sense amplifier. The former allows writing data into core-cells whereas the second is used to read their contents. These two blocks are presented in detail in Chapter II and Chapter III of this part

The core-cell (see Figure I.2) stores memory data. It is based on the latch principle, *i.e.* it is composed by two cross-coupled inverters resulting in a latch structure, and two access transistors (*Mtn3* and *Mtn4*). Data is stored as voltage levels at the two sides of the latch. A logic '1' is stored into the core-cell if node S is high and node SB is low; the opposite states on both nodes are required for storing a logic '0'. For read or write operations, the access transistors *Mtn3* and *Mtn4* are turned on when the word line is selected (its voltage goes high); thus connecting the latch to the bit lines BL and BLB.

Note that both BL and BLB lines are useful for read/write operations. The access transistors behave as transmission gates allowing bi-directional current flow between the latch and the BL and BLB lines.

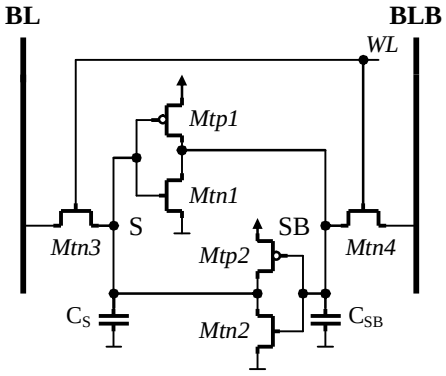

Figure I.2 – Core-cell scheme

I.1.1.2. Fault modeling

I.1.1.2.a. Fault classification [VDG00]

In this sub-section we define some terms that will be regularly used in the following. 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 functional fault model (FFM):

- a list of performed memory operations.
- 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). 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.

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 [VDG98]. Recently, another functional parameter, the output value of a read operation, has also been considered to be relevant [VDG99]. Therefore, any difference between the observed and expected memory behavior can be denoted by the following notation $< S/F/R >$. $\boldsymbol{S}$ describes the sensitizing operation sequence that sensitizes the fault. $\boldsymbol{F}$ describes the value or the behavior of the faulty core-cell; $\boldsymbol{F} \in \{0, 1, \uparrow, \downarrow, -\}$. $\boldsymbol{R}$ describes the logic output level of a read operation in case S contains read operations. 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.

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 I.3).

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 2-coupling FP or 2-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 = 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 I.3 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.

Since an FFM is defined as a set of FPs, it is expected that FFMs will inherit the properties of FPs:

- if an FFM is defined as a collection of single-cell static FPs, then the FFM is a single-cell fault model. SAF or TF are such FFM.
- If an FFM is composed by a set of two-operation FPs, then the FFM may be either a single-cell dynamic FFM or a coupling dynamic FFM. For example dynamic Read Destructive Fault (dRDF) is a single-cell dynamic FFM and Un-Restored Destructive Write Fault (URDWF) is a 2-coupling dynamic FFM.

- 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 [VDG98] and data retention faults [DEK90].

In the remaining of the thesis, we will focus on the dynamic FFM space (see right hand of Figure I.3), *i.e.* faults requiring more than one operation in the SOS.

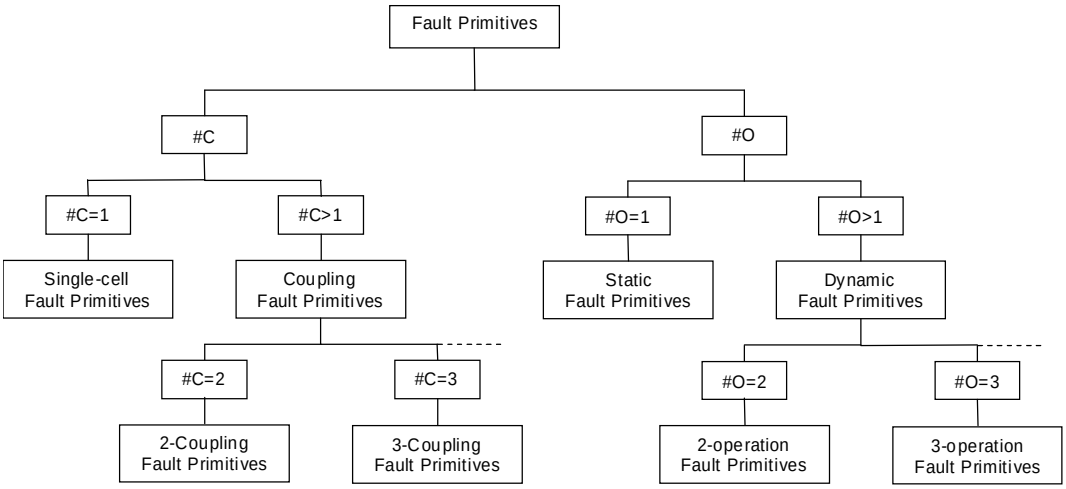

Figure I.3 – Taxonomy of fault primitives

I.1.1.2.b. Test patterns and algorithms

Remember that a memory is a particular circuit having a large quantity of internal states related to its size, *i.e.* 2^n with n the number of bits in the memory. Because of time constraints, the test of all possible internal states of memory is not possible. Currently, memories achieve more than 1Gbits of storage capacity. For instance, with a $O(2^n)$ test procedure, a 4Mbits SRAM would be tested in 500 hours. Thus, based on their regular structure and on their FFMs, researchers have developed new test methods and algorithms with a linear complexity (O(n)). Traditional memory tests include many well-known tests such as GALPAT, checkerboard, sliding diagonal, etc… [VDG98]. These test solutions are not based on fault models, such as SAF and CF, thus their quality in terms of fault coverage is difficult to be proved [VDG98]. Although simple to implement and test time advantageous, these patterns present a low fault coverage and only the SAF detection is guaranteed.

Consequently, new test methods, called March tests, have been developed. Such tests achieve a high coverage for SAF, TF or CF. March algorithms have a linear complexity

(O(n)) and more flexibility thanks to their Degree of Freedom (DOF) [NIG98], defined below. We assume the definition of a March test described by [SUK81]:

A March test consists of a finite sequence of March elements. A March element is a finite sequence of operations applied to every core-cell of memory before proceeding to the next cell. The latter can be done in either one of two address orders: an increasing (↑) address order (e.g. from address 0 to address n - 1), or a decreasing (↓) address order which is the opposite of the ↑ address order. When the address order is irrelevant the symbol ↕ is used. An operation can consist of: writing a logic '0' into a cell (w0), writing a logic '1' into a cell (w1), reading a cell with expected value '0' (r0), and reading a cell with expected value '1' (r1). Note that all operations of a March element are performed at a certain address, before proceeding to the next address.

Degrees of freedom

DOF I. The address sequences can be freely chosen as long as all addresses occur exactly once and the sequence is reversible

DOF II. The address sequence for initialization can be freely chosen as long as all addresses occur at least once.

DOF III. If the March test is built symmetrically (detects for example both SA0 and SA1 faults), the data written to the cells can be exchanged completely

DOF IV. The data within a read/write operation does not necessarily has to be equivalent for all memory addresses as long as the detection probabilities for basic faults are not affected

DOF V. The input data is not defined during read operations

DOF VI. The output data is not defined during write operations

I.1.2. Dynamic faults testing

As mentioned above, memory testing is based on the use of algorithms, especially March test algorithms. In general, the static faults are covered by a certain number of common March test algorithms. On the other hand, these algorithms are not effective for the test of the dynamic faults, which require the use of specific test sequences. Previous works done in the field of memory test algorithms targeting dynamic faults are very limited.

In [VDG02], an exhaustive set of single-cell 2-operation FPs is generated. It results in a first set of possible read/write combinations and induced faulty behavior. From this starting point, instead of taking into account all possible FPs (which may represent a too high set of possibilities), authors extracted a sub-set of 12 single-cell 2-operation dynamic FPs considering that those are the most realistic ones based on *ad-hoc* assumptions. In the same way, an exhaustive set of 2-cell 2-operation FPs is generated. It results in a second set of possible read/write combinations. Once again, a sub-set of 24 2-cell 2-operation dynamic FPs is considered as being the more realistic ones still based on *ad-hoc* assumptions.

Many studies are based on the detection of the complete set of FPs exhaustively generated like in [VDG02]. In [HAM02], two March test algorithms, March RAW1 of length 13N and March RAW of length 26N, for classes of realistic single-cell and 2-cell 2-operation dynamic faults respectively, were proposed. In [BEN05a], two March test algorithms, March AB1 of length 11N and March AB of length 22N, for the same classes as RAW1 and RAW respectively (*i.e.* realistic single-cell and 2-cell two-operation dynamic faults respectively) were proposed, thus improving the length of those proposed in [HAM02]. In [BEN05b], a March test algorithm of length 100N was proposed for detection of 2-cell dynamic faults with two operations both applied on the victim or aggressor cells. Compare to [HAM02] and [BEN05a], the sub-set of 2-cell 2-operation is enlarged. In [HAR06], authors proposed a March test algorithm of length 70N, targeting the same faults as in [BEN05b], thus improving the length by 30N of that proposed in [BEN05b]. They proposed also a March test algorithm able to deal with the overall single-cell 2-operation dynamic faults described in [VDG02].

We can thus imagine to create algorithms able to deal with x-cell y-operation dynamic faults, where $x \rightarrow n$ (n is the number of core-cells) and $y \rightarrow \infty$. Consequently, the dynamic fault class is infinite as the number of operations required for their sensitization is not limited. Based on the methodology presented above, the resulting March tests complexity becomes too high for industrial application due to huge required test time (algorithm complexity more than 100N). So, it is not possible to deal with all dynamic faults without increasing considerably the characterization time. In addition, such method is not based on a complete understanding of real defects that must appear in memory. Therefore, considering an exhaustive set of FPs bring to the consideration of improbable faulty behaviors.

Instead of considering all possible dynamic faults, a new approach consisting in first injecting actual defects and then studying the memory behavior in presence of such defects is developed. This new approach is more realistic as no assumptions on FPs are done without

understanding and verify the validity of the faulty behavior. In this approach, the memory layout is first considered and potential defective sites as well as the physical origins inducing a faulty behavior are highlighted. It has been shown in [JAM01, AZI05] that the two major types of defects that occur during the manufacturing ICs are opens and bridges defects.

The resistive-bridge defects may be due to salicide break occurring inside the core-cell. In [AZI05], authors show that such defects in the core-cell array may be the cause of dynamic faults. As results, a March test called DITEC+ has been proposed.

The significance of resistive-open defects has considerably increased in recent technologies, due to the presence of many interconnection layers and an ever-growing number of connections between each layer. In particular, in [JAM01] Intel reports that resistive-open vias are the most common root cause of test escapes in deep-submicron technologies. Hence, resistive-open defects and the faulty behavior that they involve have already been considered in the memory testing literature. With respect to the layout, these defects have been placed in correspondence of the interconnections. Based on this approach, some memory blocks have been studied, the core-cell [BOR03b, DIL04b, DIL05c, BOR05], the address decoder [SAC97, DIL04a, DIL06b] and the pre-charge circuit [DIL05b, DIL07]. As results, March algorithms dealing with dynamic faults in such blocks have been developed. Especially, authors have proposed modifications (thanks to DOFs describe above) on a well known March algorithms, the March C- (see Figure I.4). This approach seems more interesting, especially for industrial applications, as the test phase target only realistic faults. March algorithms complexity is thus reduced.

$$\updownarrow (w0) \uparrow (r0, w1) \uparrow (r1, w0) \downarrow (r0, w1) \downarrow (r1, w0) \updownarrow (r0)$$

Figure I.4 – March C- algorithm

However, few works have been done on resistive-open defects in the write driver and in the sense amplifier [ADA02]. This thesis overcomes this missing and proposes a study of these two blocks in presence of resistive-open defects.

I.1.3. Conclusion

In this chapter, we have defined the background on memory testing as well as a brief state-of-the-art on dynamic fault testing. We show that memory testing is most of time based on March test algorithms as they present a low complexity (O(n)) and are flexible thanks to

their DOFs. Based on the use of such algorithms, some studies dealing with dynamic fault testing have been done. Two main approaches are distinguishable: the first one consists in using an exhaustive set of FPs and generate specific test algorithms able to detect it. This solution induces a test time increase which may be inadequate for industrial applications as the set of FPs defining dynamic faults is infinite. The second approach consists in first injecting actual defects from layout extraction, studying the induced SRAM faulty behaviors and then generating an adapted March test algorithms. Based on this second approach, studies on core-cell, pre-charge circuit and address decoder have been published so far. In the remaining of this part, we propose to complete these works with a study of the write driver and the sense amplifier.

Chapter 2. Dynamic Faults in SRAM write drivers

In this chapter, we propose an analysis of dynamic faults induced by the presence of resistive-open defects in the write driver of SRAMs. We have inserted actual resistive-open defects in some locations of a write driver and we have performed electrical simulations in order to evaluate their effects. We have analyzed the influence of each single defect on the functional memory operations. We show that, some resistive-open defects may lead to dynamic behaviors, that can be modeled as Slow Write Driver Fault (SWDF) [VDG04], Un-Restored Write Fault (URWF) [ADA97] and Un-Restored Destructive Write Fault (URDWF). The latter has never been experienced in the past. These fault models are studied and possible March test solutions to detect them are provided. All simulations are performed on an SRAM designed with an Infineon 65nm technology.

This chapter is organized as follows: Section 1 presents the write driver fault-free functioning. Section 2 lists all possible locations of resistive-open defects in the write driver and gives the corresponding faulty behaviors. Section 3 presents a complete analysis of SWDF as well as a March test algorithm to detect such a type of fault. In the same way, Section 4 proposes an URDWF and URWF analysis. Finally, concluding remarks are given in Section 5.

I.2.1. Write driver fault-free functioning

By groups of columns in an SRAM, a write driver is used to control the true bit line (BL) and the complement bit line (BLB) during a write operation. As the two bit lines are pre-charged to Vdd before every operation, the write driver has just to act the pull down of one of the two bit lines during a write operation:

- BL for a write '0' ($w0$) operation
- BLB for a write '1' ($w1$) operation

The considered write driver structure is depicted in Figure I.5. It is composed by a write control part and a driver part. The first part receives the data that has to be written (DataIn) and the Write Enable signal (active at low level) which controls the write operation with its two outputs, named AW0 and AW1. If DataIn = 0 and the write enable signal is active, then AW0 = 1 and AW1 = 0. In that case, transistor *Mtn1* acts the pull down of BL which

corresponds to a $w0$ operation. In the same way, if DataIn = 1, AW0 = 0 and AW1 = 1, so that transistor *Mtn2* acts the pull down of BLB. It is a $w1$ operation.

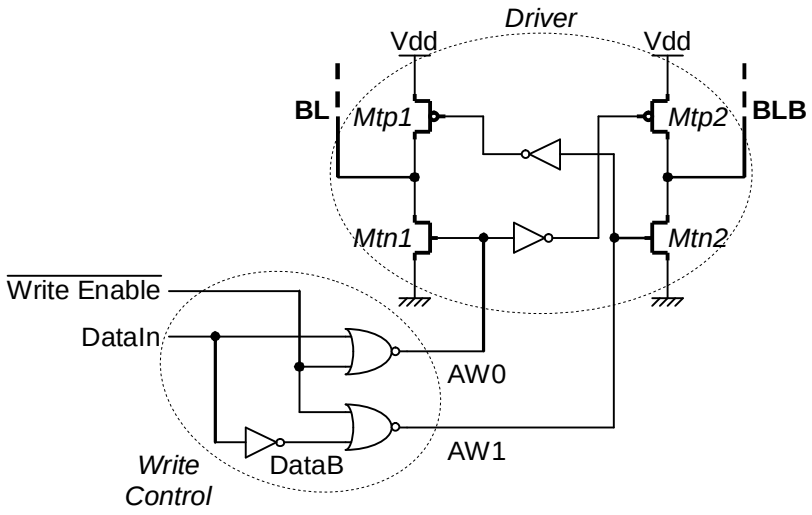

Figure I.5 – Write driver structure

Remark*: At this point, it is important to notice that, for a fault-free write driver, signals AW0 and AW1 can never be set to logic '1' at the same time.*

Waveforms presented in Figure I.6 show the correct action of the write driver during two consecutive write operations. Especially, a $w1$ operation is performed followed by a $w0$ operation on a core-cell that initially contains a logic '0'. S and SB are the state values of the selected core-cell. These waveforms were obtained for typical operating conditions, *i.e.* process: typical, voltage: 1.2V, temperature: 27°C.

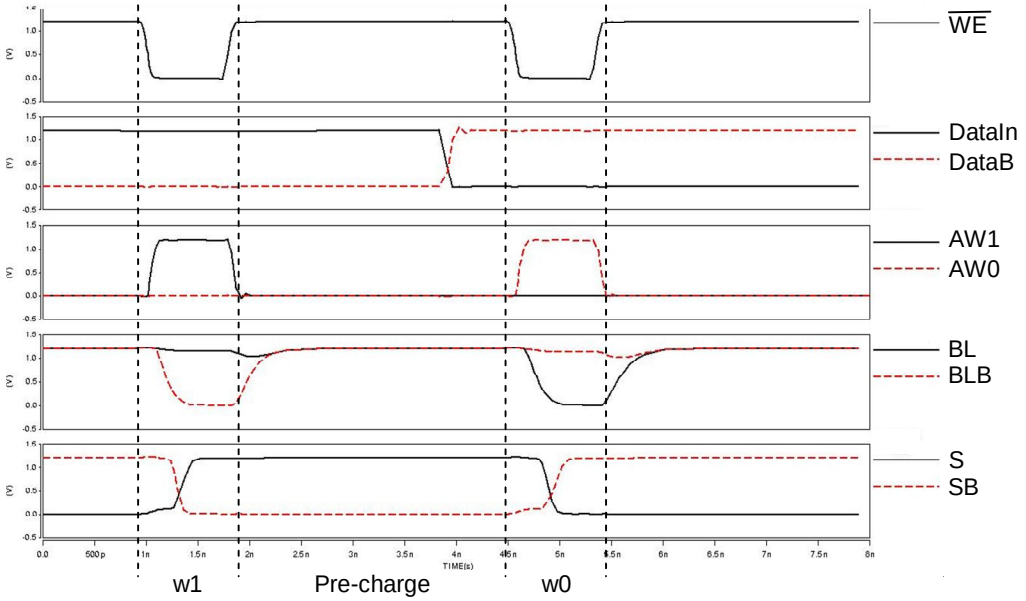

Figure I.6 – Fault-free write driver waveforms (w1, w0)

I.2.2. Resistive-open defects in the write driver

In this section, the effects induced by resistive-open defects on the normal function of the write driver circuit are analyzed. We assume the presence of only one defect for each analysis because the occurrence of multiple defects is unlikely.

As shown in Figure I.7, nine resistive-open defects (Df1 to Df9) have been placed in different locations of the analyzed write driver. We do not consider all possible locations because of the symmetry of the write driver structure. In particular, we have chosen the left part of the driver for defects Df1 to Df4. Finally, two defects (Df5 and Df6) have been considered in the inverter and three defects (Df7 to Df9) in one of the NOR gates of the write control part. Symmetric defects can be placed on the other NOR gate of the write control part and in the right part of the driver.

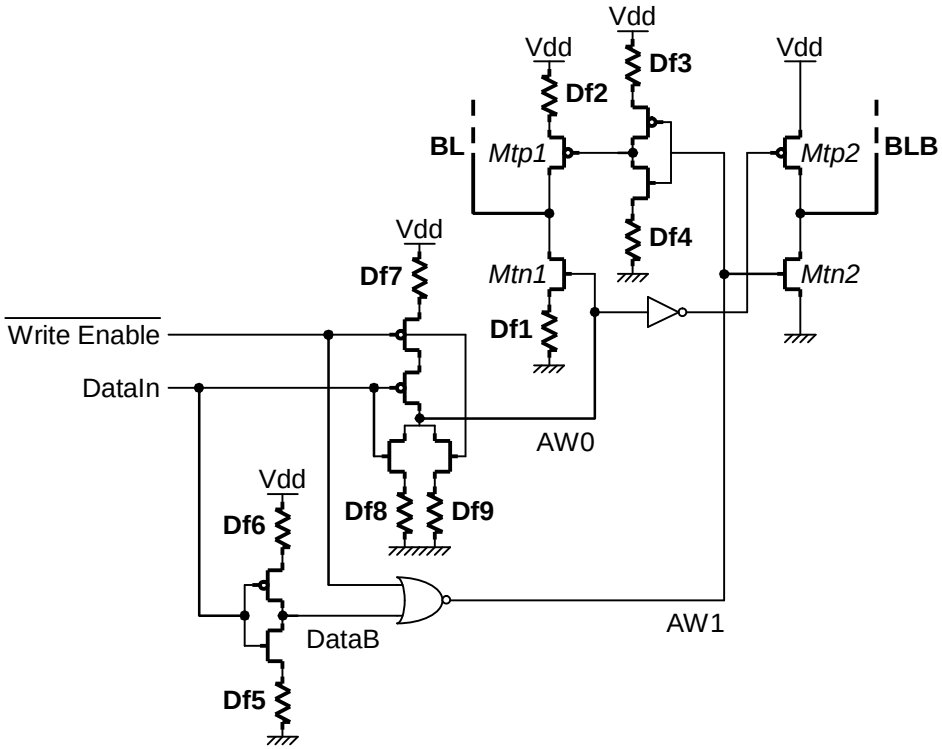

Figure I.7 – Defect injection in the write driver

I.2.2.2. Defect incidence analysis

The faulty behaviors produced by each defect in the write driver are described below.

Defect Df1: This defect produces a delay in the discharging phase of BL during the writing phases. The faulty behavior related to Df1 can be modeled by a TF. This fault is a

static fault and many classical March tests are able to detect it. The definition of such fault is provided below:

Transition Fault (TF): *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.*

Defect Df2: This defect induces a delay in the charging operation of BL during the writing phases. In presence of such defect, the pull up of node BL cannot be performed but, as the write driver has also a pre-charge circuit, the pull up is acted any how. Consequently, no faulty behavior occurs.

Defect Df3: This defect prevents to turn off of transistor *Mtp1*. Consequently, *Mtp1* is still turned on. The worst case should be if transistor *Mtn1* has to fight against transistor *Mtp2* during a $w0$ operation. However, a specific sizing is done to have the N plan (*Mtn1* and *Mtn2*) at least 5× stronger than the P plan (*Mtp1* and *Mtp2*) and hence insure the pull down of the bit line (BL for a $w0$ and BLB for a $w1$) in the time allowed for the write operation. Thus, even if *Mtn1* has to fight against *Mtp2*, the resulting level on BL is '0'. Df3 has hence no impact on the write driver functioning.

Defect Df4: This defect produces effects similar to Df2.

Defects Df5 and Df6: During a write operation, one of the two bit lines is driven to '0' and the other one remains at Vdd. However, in presence of Df5 or Df6, this operation cannot be performed, especially when there are two successive write operations with an opposite value. This faulty behavior can be modeled as Slow Write Driver Fault (SWDF):

Slow Write Driver Fault (SWDF) [VDG04]: *A write driver is said to have a SWDF if it cannot act a w0 (w1) when this operation is preceded by a w1 (w0). That results on the core-cell that does not change its data content.*

Defect Df7: This defect produces effects similar to Df1.

Defect Df8: This defect prevents the pull down of node AW0 but this action is still acted by the parallel NMOS transistor controlled by the write enable signal. Consequently, no faulty behavior is generated by such defect.

Defect Df9: A $w0$ operation can be performed by the write driver, *i.e.* AW0 node (see Figure I.7) can be set to a logic '1'. Normally, at the end of the operation, the Write Enable signal performs the pull down of node AW0. But Df9 prevents this pull down and thus node AW0 remains at logic '1' a certain time depending on the defect size. Hence, the write driver continues to perform a $w0$ even if a read operation has to be done. This faulty behavior is

modeled as an Un-Restored Destructive Write Fault (URDWF) or an Un-Restored Write Fault (URWF) (depending on the defect size):

Un-Restored Write Fault (URWF) [ADA97]: *The pull up of one of the two bit lines is not completely achieved after the state reached with a write operation. Consequently the following read operation of an opposite data in a cell belonging to the same I/O circuitry is not correctly acted.*

Un-Restored Destructive Write Fault (URDWF): *The same definition as URWF but in addition to the faulty read operation, the core-cell flips.*

I.2.2.3. Simulation set-up and results

Now we show the simulation results concerning the nine resistive-open defects analyzed in the previous sub-section. All electrical simulations of these defects have been performed with the Infineon internal SPICE-like simulator, considering at first a reference 8Kx32 Infineon 65 nm memory block, organized as an array of 512 word lines x 512 bit lines.

The whole operating environment range has been examined with the aim of determining the minimum defect size implying a faulty behavior. Hence simulations have been performed by applying a number of different test patterns and by varying the following parameters:

- Process corner: slow, typical, fast, fast n / slow p, slow n / fast p
- Supply voltage: 1.08V, 1.2V, 1.32V
- Temperature: -30°C, 27°C, 110°C
- Defect size has been swept from a few Ωs up to several MΩs.

Table I.1 presents a summary of the fault models identified for each injected resistive defect, along with the conditions for maximum fault detection, *i.e.* the minimum detected resistance value.

The first column (Dfi) indicates the defect location in the write driver with respect to Figure I.7. The second column gives the corresponding fault models. The last four columns correspond to the electrical parameters which maximize the fault detection.

Defect	Fault Model	Min Res (kΩ)	Process corner	Voltage (V)	Temp (°C)
Df1	TF	0.4	Fast	1.08	-30
Df2	-	-	-	-	-
Df3	-	-	-	-	-
Df4	-	-	-	-	-
Df5	SWDF	128	Fast	1.08	-30
Df6	SWDF	170	SF	1.32	110
Df7	TF	9.5	Fast	1.32	-30
Df8	-	-	-	-	-
Df9	URWF / URDWF	72 / 110	Slow	1.08	-30

Table I.1 – Summary of worst-case PVT corners for the defects of Figure I.7 and corresponding minimum detected resistance and fault models

As a concluding remark, we can notice that resistive-open defects in the write driver of an SRAM may be the consequence of a static fault (TF) as well as dynamic ones (SWDF, URWF / URDWF). The static fault is well known and it is detected by classical March tests. Therefore, in the next Section, we analyze the dynamic behaviors, represented by Slow Write Driver Faults and Un-Restored Destructive Write Faults.

I.2.3. Slow Write Driver Faults testing

As shown in the previous Section, SWDF can be produced by defaults Df5 and Df6. Here, we propose a complete understanding of the SRAM functioning in presence of such defects.

I.2.3.1. Detailed analysis of Df5 and Df6

During a write operation, one of the two bit lines is driven to ‘0’ and the other one remains at Vdd. However, in presence of Df5 or Df6, this operation cannot be performed, especially when there are two successive write operations with an opposite value.

On this basis, SWDFs can be defined with four FPs, which are divided in two groups.

The first group corresponds to defect Df5:

FP1: $< 1w0w1/0/- >$ A logic '1' is initially stored in the core-cell. Then, a $w0$ is acted immediately followed by a $w1$. The core-cell remains at a logic ‘0’.

FP2: $< 0w0w1/0/- >$ A logic '0' is initially stored on the core-cell. Then, a $w0$ is acted immediately followed by a $w1$. The core-cell remains at a logic ‘0’.

The second group of FPs corresponds to defect Df6:

FP3: $< 0w1w0/1/- >$ A logic '0' is initially stored on the core-cell. Then, a $w1$ is acted immediately followed by a $w0$. The core-cell remains at a logic ‘1’.

FP4: $< 1w1w0/1/- >$ A logic '1' is initially stored on the core-cell. Then, a $w1$ is acted immediately followed by a $w0$. The core-cell remains at a logic ‘1’.

As the data initially stored in the core-cell does not influence the behavior of the write driver, the following equivalences between FPs can be done:

$$FP1 \equiv FP2 \text{ and } FP3 \equiv FP4$$

Consequently, we focus only on FP1 and FP3. Note that SWDF is a dynamic fault as it requires two consecutive operations (two write operations) to be sensitized.

I.2.3.1.a. Df5 analysis

Waveforms in Figure I.8 present the faulty behavior of the memory in presence of Df5 with typical PVT conditions (Process Typ, Voltage 1.2V and Temperature 27 °C) and a defect size of about 900 kΩ.

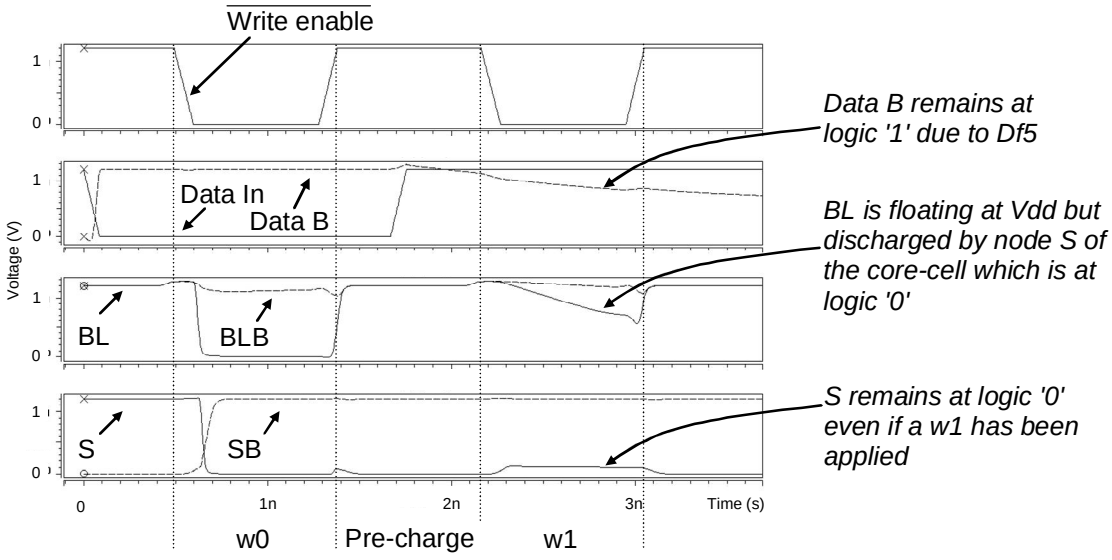

Figure I.8 – Waveforms of $< 1w0w1/1/0 >$ simulation (Df5)

The simulation starts on a core-cell that initially contains a logic '1'. We first apply a $w0$ operation. Node DataIn is set to logic '0' and node DataB is set to logic '1' before the write operation. This first write operation is correctly acted on the core-cell which switches from logic '1' to logic '0'. Then a $w1$ operation is performed. Just before this operation, DataIn is set to logic '1' but node DataB remains to logic '1'. In that case, the pull down of node DataB cannot be performed due to the presence of Df5. The two nodes AW0 and AW1 are set to logic '0'. Any write operation cannot be performed as the four transistors of the driver (*Mtp1*, *Mtn1*, *Mtp2* and *Mtn2*) are turned off. The two bit lines are floating at Vdd level. This scenario is represented in Figure I.9.

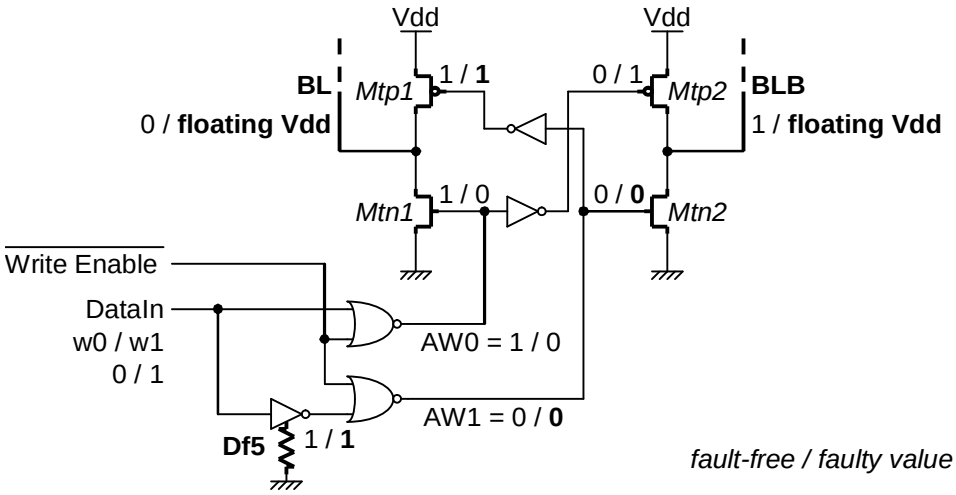

Figure I.9 – Configuration of the write driver in presence of Df5

I.2.3.1.b. Df6 analysis

Waveforms in Figure I.10 present the faulty behavior of the memory in presence of Df6 with the same operating conditions as the ones used for Df5.

The simulation starts on a core-cell that initially contains a logic '0'. We first apply a $w1$ operation. Node DataIn is set to logic '1' and node DataB is set to logic '0' before the write operation. This first write operation is correctly acted and the core-cell switches from logic '0' to logic '1'. Then, we act a $w0$ operation. Just before this operation, DataIn is set to a logic '0' but node DataB remains to a logic '0'. In that case, the pull up of node DataB cannot be performed due to the presence of defect Df6. The two nodes AW0 and AW1 are set to logic '1'. This configuration is problematic as it means that the driver has to act simultaneously a $w0$ (AW0 = 1) and $w1$ (AW1 = 1) operations. From an electrical level point of view, the four transistors of the driver are turned on. Thus, there is a resistive short between Vdd and the Gnd nodes.

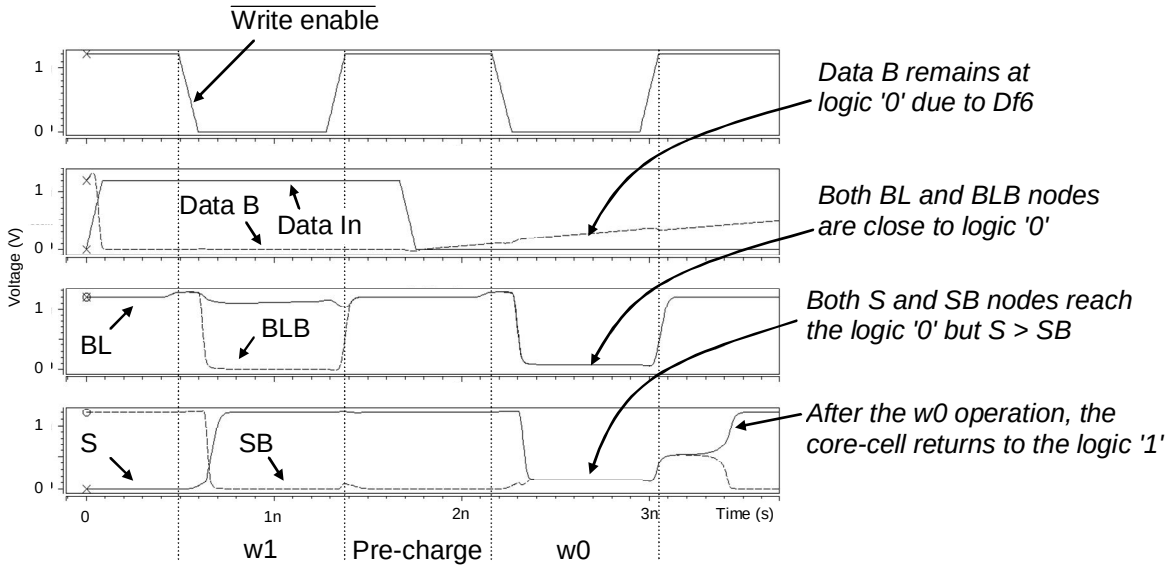

Figure I.10 – Waveforms of $< 0w1w0/0/1 >$ **simulation (Df6)**

In order to define the level of BL and BLB nodes, we must analyze the size but also the purpose of each transistors of the driver. For the same size, it is well known that NMOS transistors are stronger than PMOS transistors. For primitive gates (INV, NAND, NOR etc …), the sizing of N and P plans is done so as to balance their current driving capabilities. P plans are therefore larger than the N plans. In our case, the problem is different. The driver must act the pull down of one of the two bit lines which are equivalent to non negligible capacitances due to their length. The pull up of the two bit lines is done by the PMOS (*Mtp1* and *Mtp2*) of the driver which is helped by the pre-charge circuit. However, as previously mentioned, the N plan (*Mtn1* and *Mtn2*) is designed stronger than the P plan (*Mtp1* and *Mtp2*) insuring the pull down of the bit line (BL for a $w0$ and BLB for a $w1$) in the time allowed for the write operation. With this specific sizing, the resulting voltages on BL and BLB are then close to '0' during the $w0$ operation as seen in Figure I.10. This level on the two bit lines disturbs the core-cell content (nodes S and SB) but after the $w0$ operation, the core-cell returns to logic '1'. This scenario is represented in Figure I.11.

The two defects have the same consequences on the memory behavior although the electrical phenomena are a little bit different. The faulty behavior results in a bad write operation if it is performed after another write with an opposite data.

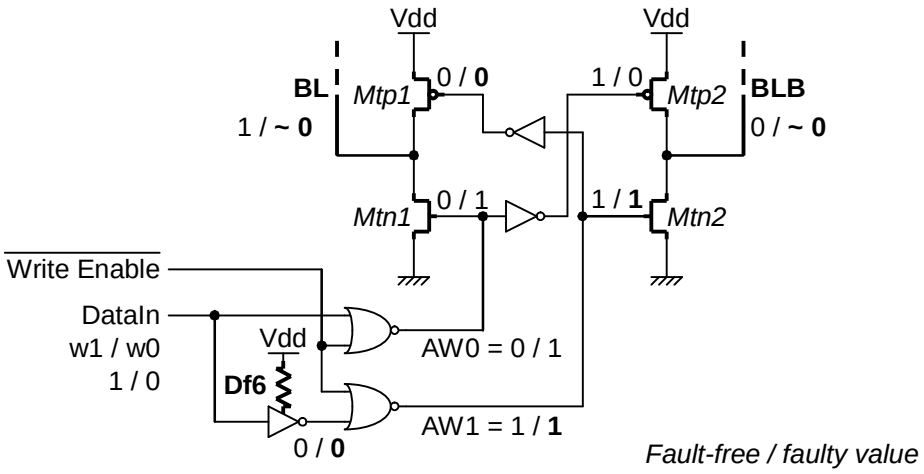

Figure I.11 – Faulty behavior of the write driver in presence of Df6

I.2.3.2. March test solution to detect SWDF

As seen in the previous sub-section, Df5 and Df6 involve a SWDF which is a dynamic fault as it requires two successive write operations to be sensitized. From the FPs presented in the previous Section, the required successive operations to detect (sensitize and observe) SWDFs is:

$$wx\ w\bar{x}\ r\bar{x}$$

where the two write operations are for sensitization of the fault and the read operation is for observation. x = 0 (resp. x = 1) corresponds to the detection of Df5 (resp. Df6). Let us first assume that these three operations must be applied on the same core-cell. From that statement, it is easy to create a specific March test to detect essentially SWDFs as presented in [VDG04]; March WDm (4N complexity) and March WDw (8N complexity). However, from a test point of view, it is more interesting to obtain a March test that covers not only SWDFs but rather a larger set of fault models. So, we have focused our study on finding possibilities to embed (with additional March elements) or find (with modifications based on the DOFs of March tests) the required succession of operations for SWDFs detection in existing March algorithms.

To do that, we have first to consider again the requirements presented above. Let us assume the basic view of an SRAM array as shown in Figure I.12 in which the write driver is shared by four columns. As the goal is to detect possible malfunction of the driver, it is not necessary to act the three operations on the same core-cell. In fact, the first write operation can be applied on one core-cell among the core-cells of the four columns. Then, it is not

necessary to act the second write on the same core-cell but, at least, act this write on a core-cell of the four columns that initially contains an opposite data to the data used for the first write operation. Of course, the read operation has to be performed on the last selected core-cell to control if the second write operation has been correctly performed. This statement makes the requirements less stringent. For example, let us assume that the first write is acted on CC_{30}. If the next write is acted on CC_{11}, then the fault is sensitized as both core-cells share the same write driver. The observation will be done when the core-cell CC_{11} will be read.

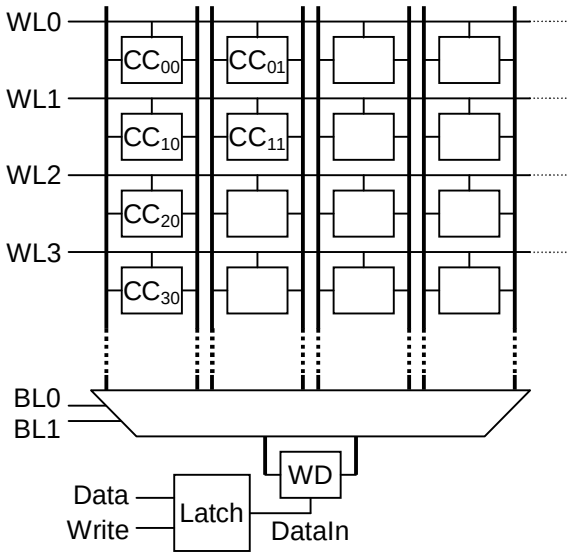

Figure I.12 – Basic view of a part of an SRAM array

In addition, we can further reduce the stringency of the required conditions to detect SWDFs. This time, we have to look deeper in the write driver structure, especially in the driver control part. It is controlled by a Write Enable signal to perform the write operation with a certain data applied on the DataIn input (see Figure I.5). This data is latched, that means, a logic '0' (logic '1') is captured in the latch for a $w0$ ($w1$) operation. An important property is that when a $w0$ ($w1$) is acted by the driver, this data (DataIn) remains stable in the latch as long as another write is not performed with the same driver. Consequently, the latch of the driver captures the first data that has to be written. Thus, it is not necessary to act immediately the second write to sensitize the write driver. Any other operation can be performed between the two write operations as long as it does not use the considered write driver. In the same way, the read operation can be preceded by read or write operations which do not change the content of the faulty core-cell. The resulting successions of operations to detect SWDFs are presented in Figure I.13.

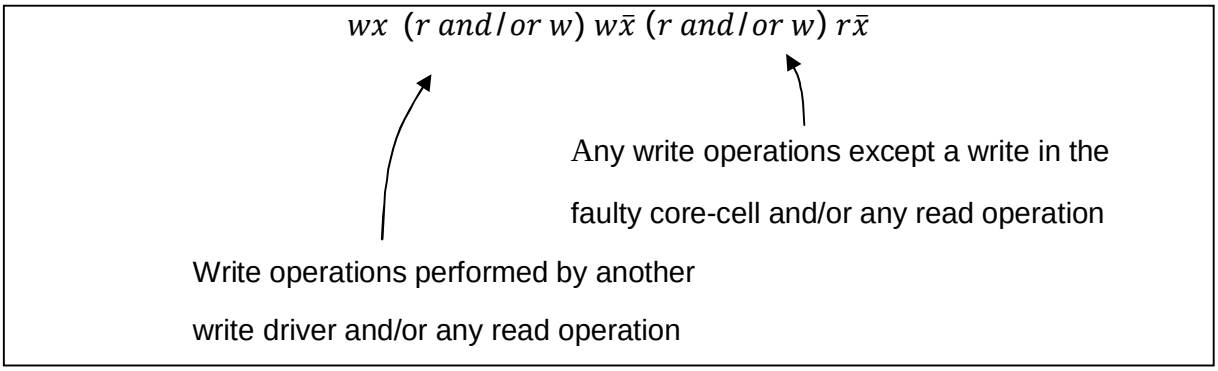

Figure I.13 – Required conditions to detect SWDFs

From these new and less stringent test conditions, we can try to find them in an existing March test. The March algorithm must have the following requirements:

- The elements of the March test have to include a $w0$ operation followed by a $w1$ operation to sensitize SWDFs induced by Df5 and a $w1$ operation followed by a $w0$ operation for those induced by Df6.
- The presence of a $r1$ operation is necessary for observation of SWDFs due to Df5 and a $r0$ operation for those induced by Df6.

These two requirements can easily be found in many March algorithms. As example, what is proposed here is to analyze if a well know March algorithm is able to detect SWDFs. In our study, we consider the March C- algorithm previously mentioned. To be perfectly, the first four elements of March C- useful for explanations are depicted in Figure I.14.

$$\underset{M0}{\updownarrow (w0)} \uparrow \underset{M1}{(r0, w1)} \uparrow \underset{M2}{(r1, w0)} \downarrow \underset{M3}{(r0, w1)} \ldots$$

Figure I.14 – March C- algorithm

We first consider the succession of M0, M1 and M2 March elements. M0 performs an initialization of the array at logic '0'. During this operation, the DataIn node of each write driver of the memory is latched at a logic '0'. Then, we act element M1 that starts by a $r0$ operation. This operation does not influence the write driver. The first time we act the $w1$ operation, the DataIn of the selected write driver is changed from logic '0' to logic '1'. This sensitizes the write drivers one after the other in SRAM. Finally, the $r1$ operation in element M2 performs the observation of possible fault effects. The succession of the three first elements (M0 to M2) allows the detection of SWDFs induced by Df5 (detected by $w0w1r1$). Table I.2 summarizes the actions of elements M0 to M2 on a simple 8 core-cell memory,

composed by two word lines, four bit lines and two write drivers as presented in Figure I.15. In order to perform the March elements, we have randomly selected the ↑ addressing order as follow:

$$Cell\ 0, 6, 1, 2, 5, 3, 7, 4$$

The ↓ addressing order is of course the reverse one.

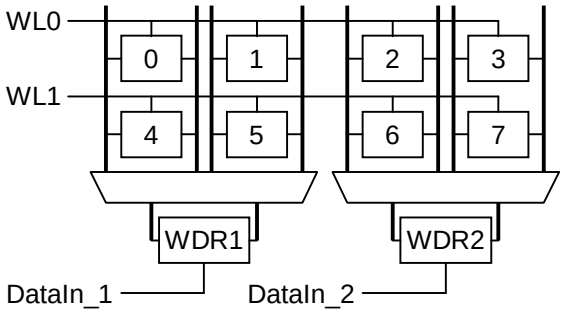

Figure I.15 – A simple 8 core-cell SRAM

Table I.2.a summarizes the action of element M0 on the SRAM depicted in Figure I.15. This element acts the initialization of the array at a logic '0'. Then, we perform element M1 (see Table I.2.b). First, cell n°0 is read and written to logic '1'. This $w1$ sensitizes the first write driver WDR1. The same occurs when the $w1$ operation is performed on cell n°6 which is the first one selected in the second group of columns. SWDFs related to Df5 are thus sensitized. Element M2 (see Table I.2.c) performs the observation by acting $r1$ operations on cell n°0 first (for WDR1), and cell n°6 next (for WDR2).

In the same way, elements M1, M2 and M3 allow the detection of SWDFs induced by Df6 (detected by $w1w0r0$). March C- is thus an efficient test algorithm to detect SWDFs in addition to faults (stuck-at, transition, coupling, etc …) initially targeted by this algorithm.

Cell n°	Element M0							
0	w0							
1			w0					
4								w0
5					w0			
DataIn_1	0	0	0	0	0	0	0	0
Cell n°2								
2				w0				
3						w0		
6		w0						
7							w0	
DataIn_2	x	0	0	0	0	0	0	0

a)

Cell n°	Element M1							
0	r0	w1						
1								
4								
5								
DataIn_1	0	1	1	1	1	1	1	1
Cell n°2								
2							r0	w1
3								
6			r0	w1				
7								
DataIn_2	0	0	0	1	1	1	1	1

Sensitization of WDR1

Sensitization of WDR2

b)

Cell n°	Element M2							
0	r1	w0						
1					r1	w0		
4								
5								
DataIn_1	1	0	0	0	0	0	0	0
Cell n°2								
2								w0
3								
6			r1	w0				
7								
DataIn_2	1	1	1	0	0	0	0	0

Observation of WDR1

Observation of WDR2

c)

Table I.2 – Application of elements M0, M1 and M2 for SWDFs detection

I.2.4. Test solution for Un-Restored Destructive Write Faults

In this section, we show that in some cases, the resistive-open defect Df9 presented in Figure I.7 may lead to a new type of dynamic behavior which has never been experienced in the past. This faulty behavior can be modeled as an Un-Restored Destructive Write Fault (URDWF). It is related to the organization of the memory and, in particular, it is the consequence of the structural dependencies that exist between the write driver and the sense amplifier. As explained previously, this faulty behavior may appear when a specific read operation is performed immediately after a specific write operation. In this section, we propose a possible March test solution to detect such type of dynamic behavior. Before-hand, we provide additional explanations on the SRAM functioning, and especially on the structural dependencies between the write driver and the sense amplifier that compose the I/O circuitry.

I.2.4.1. I/O circuitry: structural dependencies between write driver and sense amplifier

By groups of columns in an SRAM, an I/O circuitry is used to control and observe the bit line (BL) and the complement bit line (BLB) during the write and read operations. The connections of the I/O circuitry are organized as depicted in Figure I.16. An I/O circuitry is shared by some BL couples which are selected by the sub-Muxes whose activation is done by SELi signal. The functioning of the sub_Muxes is as follows:

- If SEL0 = 1 (SEL1 = … = SELm = 0) then
  - BL0 = WD = SA
  - BLB0 = WDB = SAB
- If SELm = 1 (SEL0 = … = SEL(m-1) = 0) then
  - BLm = WD = SA
  - BLBm = WDB = SAB

The selected bit lines are therefore connected to both the write driver and the sense amplifier whatever the operation (read or write). Hence, these two blocks are structurally dependent as they are always connected and disconnected to the bit lines at the same time.

Before every read or write operation, BL and BLB are pre-charged to Vdd. Write driver nodes (WD and WDB) and sense amplifier nodes (SA and SAB) are also pre-charged at Vdd by their own pre-charge circuits.

During a read operation, the sense amplifier translates the weak differential voltage between BL and BLB (ΔBL) in a full swing differential signal which is then interpreted as a digital signal to provide the logic output. The sense amplifier functioning will be detailed in the remaining of this manuscript.

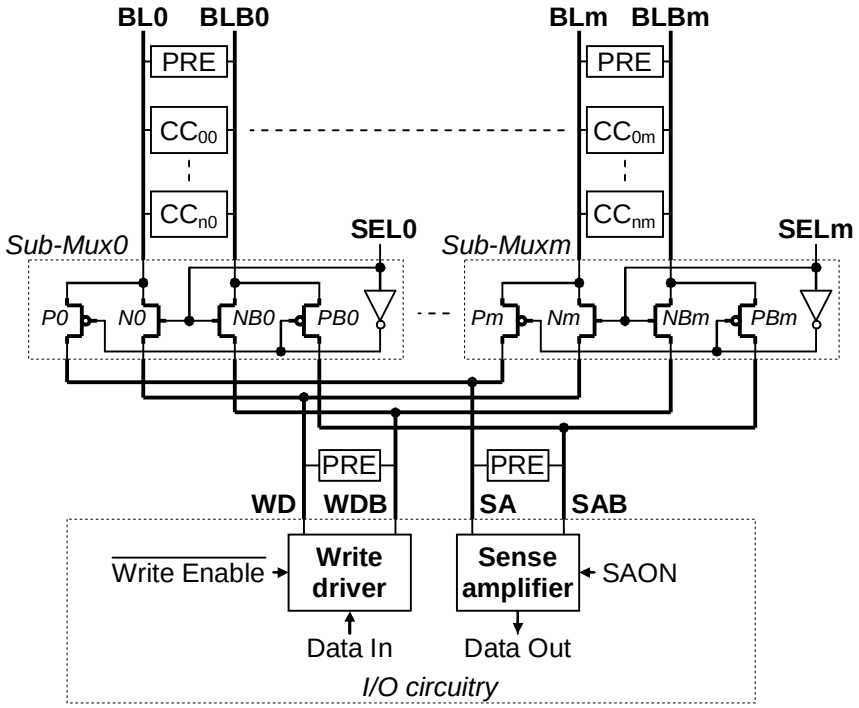

Figure I.16 – Detailed structure of the I/O circuitry

For example, let us explain what happen when there is a $w0$ operation followed by a $r1$ operation. These two operations are performed on two different core-cells (CC_A for the $w0$ and CC_B for the $r1$) belonging to the same group of column controlled by the same I/O circuitry. Figure I.17 gives the waveforms of these two operations with typical PVT conditions (typical process, 1.2V supply voltage, 27°C). Note that S (CC_A) and S (CC_B) give the electrical levels of core-cell internal nodes.

During the $w0$ operation ($w0$ on CC_A), the I/O circuitry is connected to the bit lines and the low voltage level is propagated from WD (respectively WDB) toward BL (respectively BLB), but also toward SA (respectively SAB). Note that there is a degradation of the resulting level on SA (respectively SAB) as the connection is done by a PMOS transistor which is not able to properly transfer a low voltage level (SA = V_{THp}).

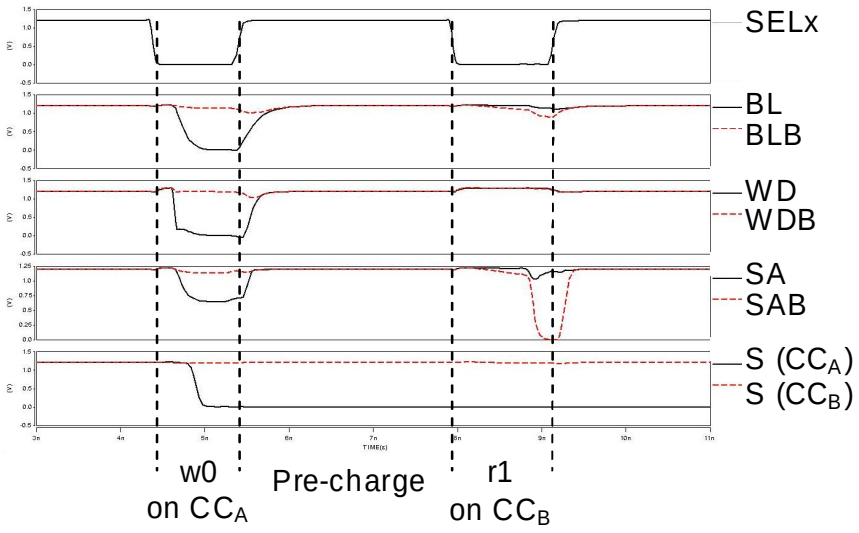

Figure I.17 – Waveforms of $w0$ and $r1$ operations

For the read operation ($r1$ on CC_B), the data is propagated from BL (respectively BLB) toward SA (respectively SAB). Note that there is no transfer from BL to WD as

- WD is at a Vdd floating level and
- the discharge of node BL is not important enough to provoke the conduction of the NMOS transistor (N0 to Nm NMOS transistors in Figure I.16) and hence insure the connection between BL and WD.

Explanation can be provided by notice that NMOS transistors are in a sub threshold functioning mode, as mentioned by *Eq. I.1*.

$$V_{gs} = Vdd - (Vdd - \Delta BL) = \Delta BL < V_{THn} \qquad \textit{(Eq. I.1)}$$

At the end of the read operation, the sense amplifier is activated to provide the logic data output; a logic '1' in our example.

I.2.4.2. URDWF analysis

In this sub-section, we detail the behavior of the write driver in presence of defect Df9. We first provide a FFM of the faulty behavior by using FPs previously defined. Next, we use electrical measurements to analyze the impact of Df9 on the behavior of the memory. As shown in Table I.1, Df9 may induce two different dynamic behaviors, either a standard URWF or a URDWF. From this statement, we provide comparisons of both URDWF and URWF. Finally, we propose a possible March test solution to detect URDWF and URWF.

I.2.4.2.a. Functional fault modeling

In presence of Df9, an Un-Restored Destructive Write Fault may occur. A $w0$ operation can be performed by the write driver, *i.e.* AW0 node (see Figure I.5) can be set to a logic '1'. Normally, at the end of the write operation, the Write Enable signal performs the pull down of node AW0. But Df9 prevents this pull down and thus node AW0 remains at logic '1' a certain time depending on the defect size. Hence, the write driver continues to perform a $w0$ even if a read operation has to be done.

Based on this description, an URDWF can be defined with four FPs, which are divided in two groups. The first group corresponds to defect Df9:

FP1: $< 1\boldsymbol{w0}, \boldsymbol{1r1/0/0} >$ A $w0$ is performed on a core-cell containing a logic '1'. Then, a $r1$ is performed in another core-cell belonging to the same group of column. This read operation makes the core-cell flipping from a logic '1' to a logic '0'.

FP2: $< 0\boldsymbol{w0}, \boldsymbol{1r1/0/0} >$ same as FP1, but this time, the $w0$ is performed on a core-cell containing a logic '0'.

The second group of FPs corresponds to the opposite defect placed in the pull down of the other NOR gate of the control part of the write driver.

FP3: $< 1\boldsymbol{w1}, \boldsymbol{0r0/1/1} >$ A logic '1' is initially stored in a core-cell. Then a $w1$ is acted; a logic ‘0’ is stored in another core-cell belonging to the same group of column; then a $r0$ is acted in this cell. This one flips to a logic ‘1’.

FP4: $< 0\boldsymbol{w1}, \boldsymbol{0r0/1/1} >$ A logic '0' is initially stored in a core-cell. Then a $w1$ is acted; a logic ‘0’ is stored in another core-cell belonging to the same group of column; then a $r0$ is acted in this core-cell. This one flips to a logic ‘1’.

As the data initially stored in the cell does not influence the behavior of the write driver, the following equivalences between FPs can be done:

$$\text{FP1} \equiv \text{FP2 and FP3} \equiv \text{FP4}$$

Furthermore, as the electrical faulty behaviors observed by applying the SOS of FP1 (in presence of Df9) and the SOS of FP4 (in presence of the opposite defect placed in the pull down of the other NOR gate) are equivalent, with opposite data to be written and read, the analysis of one of those is sufficient for a complete study of URDWFs.

I.2.4.2.b. Electrical simulations with Df9

Waveforms in Figure I.18 present the resulting faulty behavior of the memory in presence of Df9 with typical PVT conditions (typical process, 1.2V supply voltage, 27°C) and a defect size of about 500 kΩ.

The simulation starts on two different core-cells (CC_A and CC_B) belonging to the same group of columns controlled by the same I/O circuitry, both initially containing a logic '1'. We first apply a $w0$ operation on CC_A. The pre-charge circuits are switched off. Node WD drives the '0' through BL to fight against the core-cell that contains a logic '1'. This means that the NMOS transistor (*Mtn1* in Figure I.5) has to be strong enough to impose the '0' on BL. The $w0$ operation is correctly performed on CC_A that flips from a logic '1' to a logic '0'. Then, the pre-charge circuits are switched on. PMOS transistors composing the pre-charge circuits are normally strong enough to drive lines BL, BLB, WD, etc. which are equivalent to capacitances. However, these PMOS transistors are much less stronger than the NMOS transistors (*Mtn1* and *Mtn2* in Figure I.5) of the write driver. These different strengths between transistors composing the write driver and the pre-charge circuits make that node WD still remains at '0' during the pre-charge operation in presence of defect Df9. In this case, we can say that the $w0$ operation still remains active (see Figure I.18).

Afterward, the second core-cell CC_B is selected for a $r1$ operation. In order to explain the faulty behavior observed, it is important to analyze the functioning of the sense amplifier. It allows to take a decision depending on the core-cell content (logic '0' or '1'). If there is an erroneous differential voltage between BL and BLB during the read operation, the sense amplifier badly translates this differential voltage. In presence of Df9, the fact that node WD remains at '0' makes that the differential voltage is incorrect and the $r1$ operation is erroneous. As seen in Figure I.18, node SA is at Vdd and node SAB reaches '0', thus meaning that CC_B is read as containing a logic '0' and not a logic '1'. It is also important to notice that, as node WD remains at '0' during the read operation, this level performs a $w0$ on CC_B thus inducing a flipping of the core-cell from a logic '1' to a logic '0'.

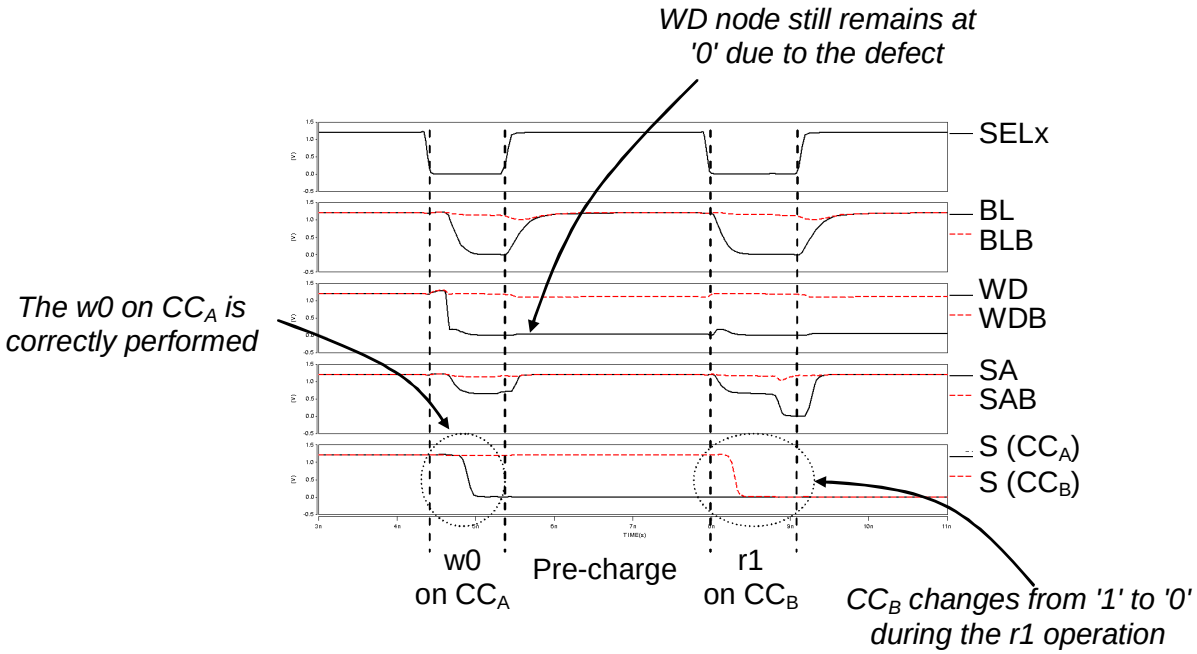

Figure I.18 – Waveforms of $< 1w0, 1r1/0/0 >$ simulation (Df9)

To summarize the effect of Df9, we can say that the $r1$ operation on CC_B has two effects related to the fact that the write driver continues to perform a $w0$ during this read operation. First, the sense amplifier provides the data given by the write driver - a logic '0' in our case. Secondly, CC_B is written to a logic '0'. So, the $r1$ operation is seen as a $w0$ operation.

I.2.4.2.c. URDWF vs. URWF

As shown previously, an URDWF may occur in presence of defect Df9. Such a faulty behavior is observed for specific write/read operations but also for a certain range of defect size (see Column 5 in Table I.1) denoted as border 2 in Figure I.19. If Df9 has a size lower than border 2 but higher than border 1, an URWF occurs. This time, there is no destruction of the data initially stored in the core-cell to be read.

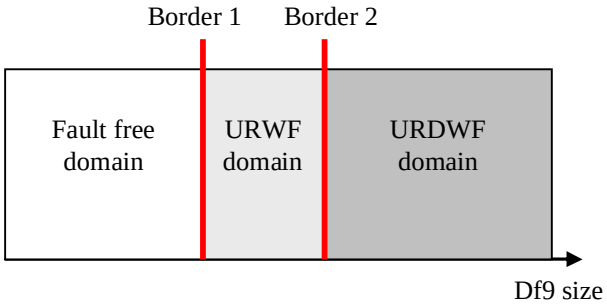

Figure I.19 – Fault type vs. defect size

Once again, let us consider two core-cells (CC_A and CC_B) to the same group of columns controlled by the same I/O circuitry. Both cells initially contain a logic '1'. Waveforms in Figure I.20 present the faulty behavior of the memory in presence of Df9 with typical PVT conditions (typical process, 1.2V supply voltage, 27°C) and a defect size of about 100 kΩ.

The $w0$ operation performed on CC_A is correctly acted as the core-cell flips from a logic '1' to a logic '0'. Then, the pre-charge circuits of the core-cells are switched on. Compared to Figure I.18, this time node WD is not at '0' but rather is increasing. This is due to the fact that the NMOS transistor (*Mtn1* in Figure I.5) is not fully saturated due to a lower defect size. Thus, it fights against the PMOS transistors of the pre-charge circuit. Then, at the beginning of the $r1$ operation performed on CC_B, the remaining voltage level on node WD is not low enough to induce the faulty swap of the core-cell. On the other hand, node WD remains at a voltage level which is low enough hence inducing that the sense amplifier badly translates the faulty differential voltage. This is shown in Figure I.20 where we can see that cell CC_B does not flip (node S of CC_B still remains at a logic '1') but the logic data output given by the sense amplifier is a logic '0' (node SA remains close to Vdd and node SAB is at '0').

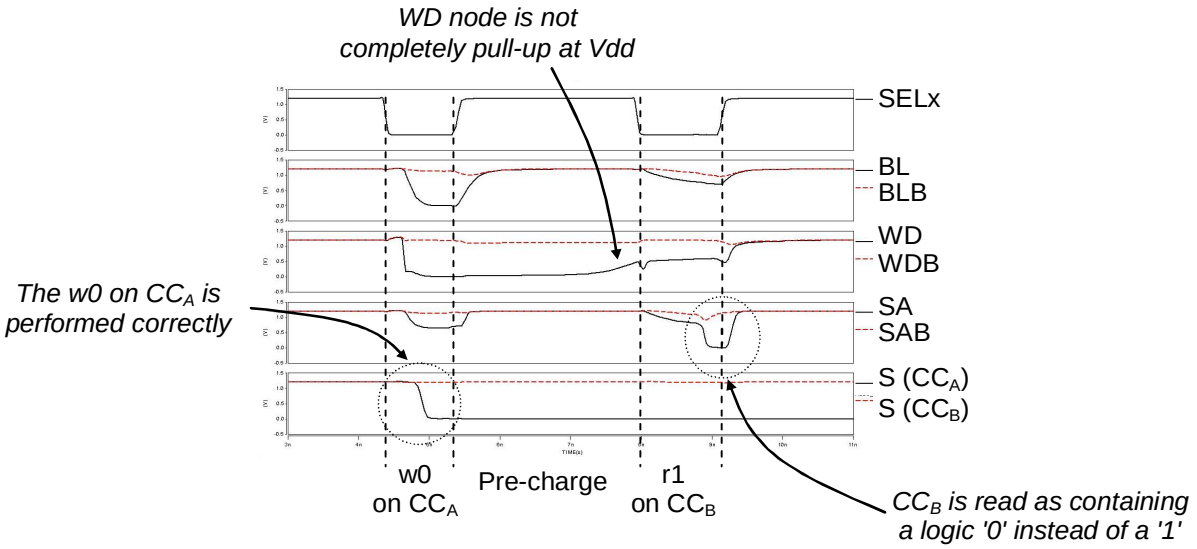

Figure I.20 – Waveforms of $< 1w0, 1r1/1/0 >$ simulation (Df9)

This electrical study shows that depending on the size of Df9, the faulty behavior can be modeled as an URWF or an URDWF. The next section provides a test solution for both fault models.

I.2.4.2.d. March test solution

As seen previously, Df9 may involve an URDWF or an URWF depending on its size. Both fault models require the same sequence of operations to be detected (sensitized and observed). This sequence is defined as follows:

$$wx\ r\bar{x}$$

where both operations have to be performed on two distinct core-cells controlled by the same I/O circuitry.

A study of URWF detection has already been done in [DIL05b]. This study shows that the March C- algorithm (see Figure I.14) with a column after column addressing order is able to detect URWFs. This particular addressing order is allowed once again by the DOFs of March tests. As the detection conditions of URWFs and URDWFs are the same, the March C- algorithm is also able to detect URDWFs.

I.2.5. Conclusion

In this chapter, we have analyzed and characterized the effects of resistive-open defects that may occur in the write driver of SRAMs. We have found that some defects do not disturb the memory behavior, some others involve a TF, and two defects in the write control part induce a Slow Write Driver Fault (SWDF). This fault prevents the write control part to correctly decide between $w0$ and $w1$ operations. By performing electrical simulations with the 65nm Infineon technology, we have evaluated the influence of these defects and show that SWDFs can easily be detected by a standard March algorithm namely the March C-.

Moreover, we have shown that a resistive-open defect may lead to a new type of dynamic behavior which has never been experienced in the past. This faulty behavior has been modeled as an Un-Restored Destructive Write Fault (URDWF). Such fault is a consequence of the structural dependencies that exist between the write driver and the sense amplifier, and appears when a specific read operation is performed immediately after a specific write operation. We have performed electrical simulations to give a complete understanding of such a faulty behavior and to highlight differences with the standard Un-Restored Write Fault (URWF) model.

Chapter 3. Dynamic faults in SRAM sense amplifiers

In this chapter, we present an analysis of dynamic faults induced by the presence of resistive-open defects in the sense amplifier of SRAMs. The validation of this work is done with a SRAM designed in 65nm technology. We have inserted resistive-open defects in some locations of a sense amplifier and we have performed electrical simulations in order to evaluate their effects. We have analyzed the influence of each single defect on the functional memory operations. We show that some resistive-open defects may lead to a new type of dynamic behavior which has never been experienced in the past. This faulty behavior can be modeled by dynamic two-cell Incorrect Read Faults of two different types (d2cIRF1 and d2cIRF2). Such fault models represent failures in the sense amplifier which prevent it to do its function, i.e. a read operation. The main difference between them is that d2cIRF1 prevents all read operations whereas d2cIRF2 prevents only a single type of read operation (either $r0$ or $r1$). As explained in this chapter, these faulty behaviors may appear when a specific sequence of read operations is performed. To complete our study, we propose a possible March test solution to detect such fault models.

This chapter is organized as follows. Section 1 presents the sense amplifier fault-free functioning. Section 2 lists all possible locations of resistive-open defects in the sense amplifier and gives the corresponding faulty behaviors. Section 3 presents a complete analysis of d2cIRF1 as well as a March test algorithm to detect such a type of fault. In the same way, Section 4 deals with the d2cIRF2 fault model. Finally, conclusions are given in Section 5.

I.3.1. Sense amplifier description

In this section, we describe the structure of each sense amplifier in the I/O circuitry. We first provide a global view of the memory including the I/O circuitries and then we detail the sense amplifier fault-free operation.

I.3.1.1. Sense amplifier within the I/O circuitry

As previously mentioned, an I/O circuitry, composed by write drivers and sense amplifiers, is used to control or observe the bit line (BL) and the complement bit line (BLB) during the write and read operations of a given core-cell. A global view of the memory

structure is presented in Figure I.21 in which we have only represented sense amplifiers (write drivers are not represented for the sake of clarity). From Figure I.21, it is important to notice that each sense amplifier has its own pre-charge circuit which is activated at the same time as the bit line pre-charge circuits.

As shown in Figure I.21, a sense amplifier is shared by several BL couples. A BL couple is selected by the signal SEL_{BLx}. During a read operation, the bit line voltage levels of selected columns are propagated towards each SA_i and SAB_i nodes ($0 \leq i \leq k$). Then, the sense amplifier corresponding to the targeted core-cell is activated by its signal $SAON_i$ (all the others remaining off). The outputs z_i and zb_i of this sense amplifier control the data output circuitry. This block generates the logic output data (Data_out). At this point, it is important to notice that the data output circuitry is shared by one or more sense amplifiers. In some SRAM configurations, two sense amplifiers can share the same data output circuitry. In some others, four sense amplifiers can share the same data output circuitry. These different possible memory configurations will be used later on in the Section to explain the d2cIRF fault model and to provide the March algorithm that can be used to detect this fault model. Note also that several data output circuitries are normally embedded in an SRAM depending on its size and structure.

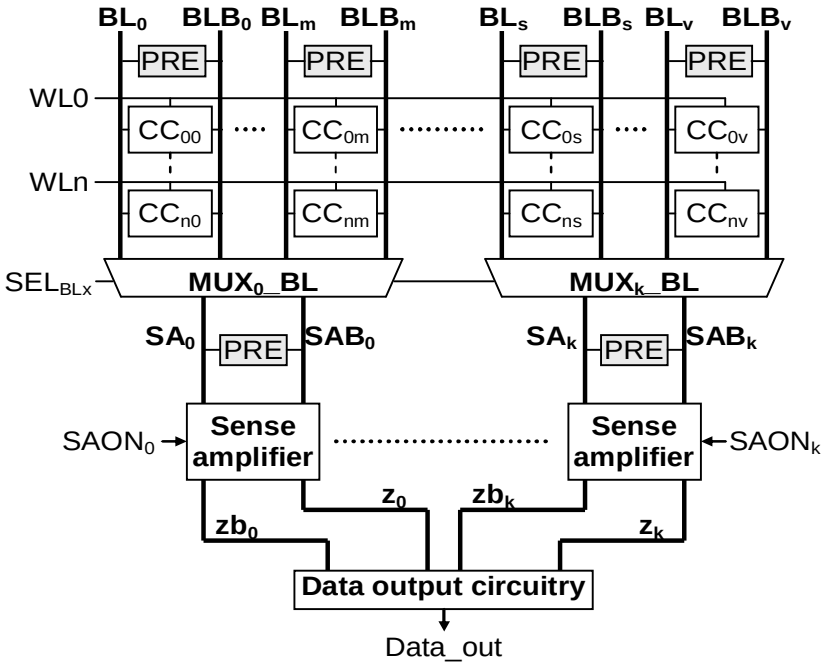

Figure I.21 – Memory structure scheme

I.3.1.2. Sense amplifier fault-free operation

The transistor view of the considered sense amplifier is presented in Figure I.22. As previously mentioned, before every read operation, BL and BLB as well as SA and SAB are pre-charged at Vdd. A read operation begins with the selection of the targeted core-cell. This access time allows one of the two bit lines (BL for a $r0$, BLB for a $r1$) to be discharged of about 100mV.

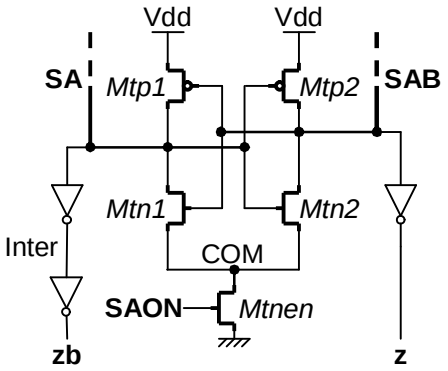

Figure I.22 – Sense amplifier scheme

The second step consists in activating the sense amplifier in order to translate this weak differential voltage between BL and BLB (ΔBL = BL - BLB = SA - SAB) in a full swing differential signal which is then interpreted as a digital signal by the data output circuitry:

- ΔBL ~ + 100mV ($r1$) ⇒ SA = 1, SAB = 0
- ΔBL ~ - 100mV ($r0$) ⇒ SA = 0, SAB = 1

At the beginning of a read operation, the two nodes SA and SAB can be interpreted as a logic '1' level signal that turns on the two NMOS transistors (*Mtn1* and *Mtn2* in Figure I.22), thus helping the discharge of the two nodes. However, the node with a lower voltage value (SA for a $r0$, SAB for a $r1$) discharges faster than the other one, thus turning on the corresponding PMOS transistor (*Mtp2* for a $r0$, *Mtp1* for a $r1$).

In summary, for a read performed on a core-cell belonging to the group i ($0 \leq i \leq k$) of core-cells controlled by the same sense amplifier, we finally have:

- for a $r0$: $SA_i = 0$ and $SAB_i = 1$ and: $z_i = 0$ and $zb_i = 0$
- for a $r1$: $SA_i = 1$ and $SAB_i = 0$ and: $z_i = 1$ and $zb_i = 1$

Note that all the others SA_j and SAB_j ($j \neq i$) nodes remain at Vdd as their sense amplifiers are disabled. Consequently, all the other z_j remain at a logic '0' and the zb_j remain at logic '1'. Then, the data output circuitry interprets the z and zb signals to provide the logic

output data (see Figure I.21). The structure of the data output circuitry is generally a latch. In our memory structure, it is not only a latch but it also used a specific and confidential control logic. Nevertheless, we report in Table I.3 the truth table representing the logic behavior of this data output circuitry.

z	zb	Data_out
0	0	0
1	0	Memory state
0	1	Memory state
1	1	1

Table I.3 – Truth table of the data output circuitry

For a $r0$ operation on a core-cell belonging to the group i, z_i and zb_i are at the logic '0' value, thus implying Data_out to be pulled down. For a $r1$ operation, both z_i and zb_i are at the logic '1' value, implying Data_out to be pulled up. Note that when no read operation is performed or during the pre-charge operation, SA and SAB remains at Vdd, thus implying z = 1 and zb = 0. With such a configuration, the Data_out signal remains stable at the logic data stored previously ("Memory state" in Table I.3).

Waveforms presented in Figure I.23 show the correct operation of a sense amplifier during two consecutive read operations. Especially, we perform a $r0$ followed by a $r1$ on two different core-cells (CC_A and CC_B) sharing the same sense amplifier. $S(CC_A)$ and $S(CC_B)$ are the state values of each core-cell. These waveforms were obtained from typical operating conditions, *i.e.* process: typical, voltage: 1.2V, temperature: 27°C.

The simulation starts with a $r0$ operation performed on CC_A. BL node is discharged and BLB node remains at Vdd. The same behavior appears on nodes SA and SAB. Then, the signal SAON is activated and the sense amplifier detects this weak differential voltage between SA and SAB. SA is fully discharged and SAB remains at Vdd, so that nodes z and zb are set to logic '0'. With such logic values, node Data_out is pulled down (*c.f.* Table I.3).

Then, pre-charge circuits are switched on. All the lines (BL, BLB, SA and SAB) are therefore forced to Vdd. We can also note that Data_out remains stable at logic '0', which corresponds to the last stored data (*c.f.* Table I.3).

The next operation is a $r1$ performed on CC_B. This time, BL node remains at Vdd while BLB is discharged. When the SAON signal is activated, the sense amplifier detects this weak

differential voltage that makes SA remaining at Vdd and SAB fully discharged at '0'. Then, nodes z and zb are set to Vdd, thus implying the pull up of node Data_out.

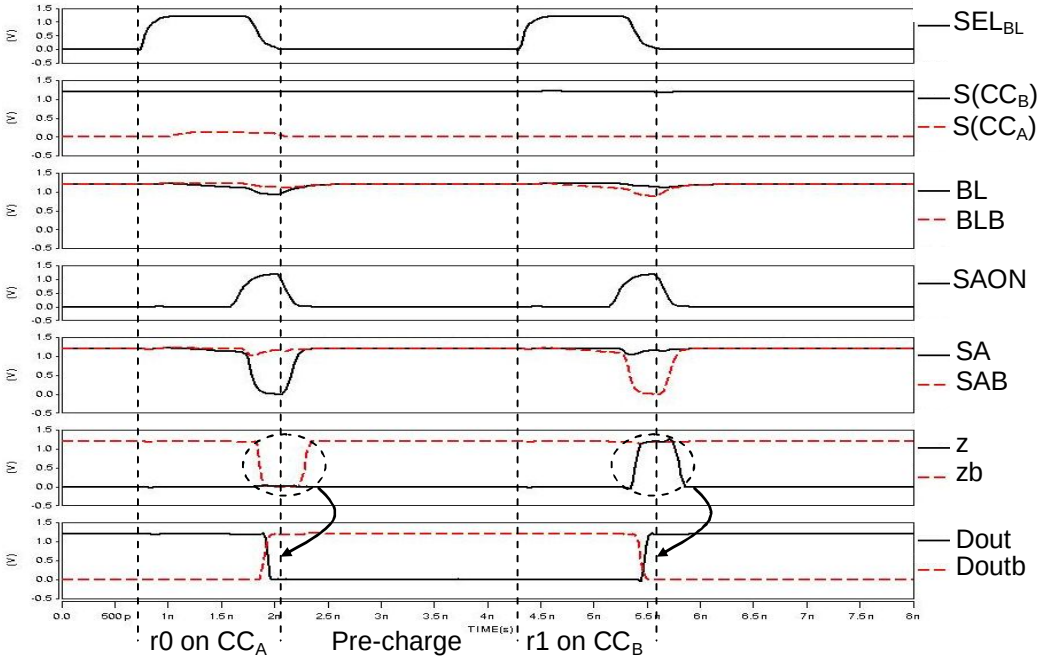

Figure I.23 – Fault-free data output circuitry waveforms ($r0$, $r1$)

Note that if the two read operations are performed on core-cells connected to two distinct sense amplifiers sharing the same data output circuitry, two distinct SAON signals and two different couples (z, zb) will be involved in the definition of the Data_out signal.

I.3.2. Resistive-open defects in the sense amplifier

In this section, we summarize the effects induced by resistive-open defects on the normal functioning of the sense amplifier.

As shown in Figure I.24, nine resistive-open defects (Df1 to Df9) have been placed in different locations of the sense amplifier. We do not consider all possible locations because of the symmetry of the structure.

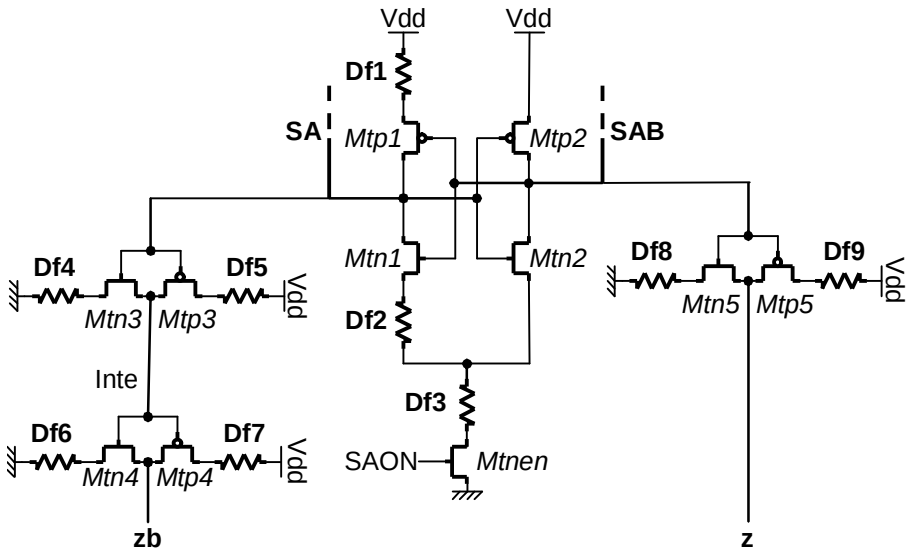

Figure I.24 – Defect injection in the sense amplifier

I.3.2.2. Defect incidence analysis

Now we detail the faulty behavior as well as the attached fault models that the defects may induce in the memory sense amplifier.

Defect Df1: In presence of Df1, the pull up of node SA cannot be performed but, as the sense amplifier has also a pre-charge circuit, the pull up is acted anyhow thus masking the effect of Df1.

Defect Df2: In presence of Df2, a read operation provides the opposite data than that stored in the targeted core-cell. In our case, a logic '1' is observed when we perform a $r0$ operation as the pull down of node SA cannot be done. This faulty behavior can be modeled as Incorrect Read Fault (IRF), which definition is:

Incorrect Read Fault (IRF): *A core-cell is said to have an IRF if a read operation performed on the cell returns an incorrect logic value, while keeping the correct stored value in the cell.*

Defect Df3: During a read operation, SA (for a $r0$) or SAB (for a $r1$) node is normally driven to '0' when the sense amplifier is activated by its SAON signal. However, in presence of Df3, this operation cannot be performed as the sense amplifier remains disabled. Then, the data output circuitry does not change its value and gives the logic data previously stored. Such faulty behavior is modeled as dynamic 2-cell Incorrect Read Fault type 1 (d2cIRF1), and the definition is as follows:

dynamic two-cells Incorrect Read Fault type 1 (d2cIRF1): *A sense amplifier is said to have a d2cIRF1 if it is unable to read any value. So, the read data value at the output is the one previously stored in the data output circuitry. This is a two-cell fault model as it requires two read operations on two distinct core-cells.*

Defect Df4 to Df9: These defects prevent the pull up or the pull down of nodes z and zb. Two successive specific read operations are therefore not possible. This faulty behavior is modeled as dynamic 2-cell Incorrect Read Fault type 2 (d2cIRF1), and the definition is as follows:

dynamic two-cells Incorrect Read Fault type 2 (d2cIRF2): *A sense amplifier is said to have a d2cIRF2 if it is only able to perform a $r0$ or $r1$ operation As for d2cIRF1, this is a two-cell fault model as it requires two read operations on two distinct core-cells.*

I.3.2.3. Simulation set-up and results

In this sub-section, we show the simulation results concerning the nine resistive-open defects analyzed in the previous sub-section. Once again, the simulations have been performed with a Spice-like simulator provided by Infineon, with a 65nm technology.

The whole operating environment range has been examined with the aim of determining the test conditions which maximize the fault detection probability. Hence simulations have been performed by applying a number of different test patterns and by varying the following parameters:

- Process corner: slow, typical, fast, fast n / slow p, slow n / fast p
- Supply voltage: 1.08V, 1.2V, 1.32V
- Temperature: -30°C, 27°C, 110°C
- Defect size has been swept from a few Ωs up to several MΩs.

Table I.4 presents a summary of the fault models identified for each injected resistive defect, along with the conditions for maximum fault detection, *i.e.* the minimum detected resistance value. The first column (Dfi) indicates the defect location in the sense amplifier with respect to Figure I.24. The attached fault models are given in the second column. Finally, the last four columns correspond to the electrical parameters which maximize the fault detection.

Defect	Fault Model	Min Res (kΩ)	Process corner	Voltage (V)	Temp (°C)
Df1	-	-	-	-	-
Df2	IRF	0.35	Fast	1.32	-30
Df3	d2cIRF1	1.8	Fast	1.32	-30
Df4	d2cIRF2	140	Slow	1.08	-30
Df5	d2cIRF2	20	Fast	1.32	-30
Df6	d2cIRF2	15	Fast	1.32	-30
Df7	d2cIRF2	150	Fast	1.32	110
Df8	d2cIRF2	140	Slow	1.08	-30
Df9	d2cIRF2	20	Fast	1.32	-30

Table I.4 – Summary of worst-case PVT corners for the defects of Figure I.24 and corresponding minimum detected resistance and fault models

As a concluding remark, we can notice that resistive-open defects in the sense amplifier of an SRAM can be modeled by a static fault (IRF) as well as dynamic ones (d2cIRF type 1 and type 2). The next Sections are dedicated to the study of these dynamic faults.

I.3.3. d2cIRF1 analysis

In this section, we detail the behavior of the sense amplifier affected by a d2cIRF1. We first provide a FFM of the faulty behavior by using FPs. Next, we present electrical measurements to analyze the impact of a d2cIRF1 on the SRAM. Finally, we propose a possible March test solution to detect this FFM.

I.3.3.1. Functional fault modeling

As mentioned in sub-section I.3.1.1, there exist several memory configurations that differ by the number of sense amplifiers sharing the same data output circuitry. However, we have to provide a generic FFM independently of the memory configuration.

In presence of Df3, a d2cIRF1 may occur depending on the defect size. During a read operation, SA (for a $r0$) or SAB (for a $r1$) (see Figure I.22) node is normally driven to ‘0’ when the sense amplifier is activated by its SAON signal. However, in presence of Df3, this operation cannot be performed as the sense amplifier remains disabled. Then, the data output

circuitry does not change its value and gives the logic data previously stored. At this point the question is: how to highlight this faulty behavior?

A straightforward solution consists in initializing the data output circuitry by performing a read operation with a given sense amplifier (Sense_1). We denote this operation as rx with $x \in \{0,1\}$. The data output circuitry is therefore initialized at the x logic value. Then, we select another sense amplifier (Sense_2) sharing the same data output circuitry and we perform a read operation with an opposite data, *i.e.* this operation is denoted as $r\bar{x}$. If Sense_2 is affected by a d2cIRF1, it cannot perform any read operation, thus meaning that the data output circuitry will remain stable at x instead of providing a $\bar{x}$ logic value. The fault is therefore sensitized and observed.

Such a test solution is only valid when there are two or more sense amplifiers sharing the same data output circuitry. However, it does not work if there is only one sense amplifier per data output circuitry. So, a solution to be independent of the memory configuration consists in performing the two read operations, rx and $r\bar{x}$, on the same sense amplifier. This time, the Data_out node is not initialized but remains stable at a constant logic value if the targeted sense amplifier is affected by a d2cIRF1.

Based on these descriptions, a d2cIRF1 can be defined with a single FP. As previously explained, a FP is denoted as $< \boldsymbol{S/F/R} >$. $\boldsymbol{R}$ takes generally $\{0,1,-\}$, where '-' is used when no read operation is required for the sensitized operation sequence S. An important point is that in our case, we need another symbol to represent the fact that the data output value does not change during every operation of S. This symbol is denoted as 'c' ('c' stands for constant). From this notation, we finally obtain a single FP for d2cIRF1:

> **FP**: $< xrx, \bar{x}r\bar{x} \,/\, \bar{x} \,/\, c >$ A rx is performed on a first core-cell. Then, a $r\bar{x}$ operation is performed with the same sense amplifier in another core-cell. The node Data_out still remains at a constant logic value 'c' during both read operations.

I.3.3.2. Electrical simulations with Df3

Waveforms in Figure I.25 present the faulty behavior of the memory in presence of Df3. They were obtained with typical PVT conditions (typical process, 1.2V supply voltage, 27°C) and a defect size of about 10 kΩ.

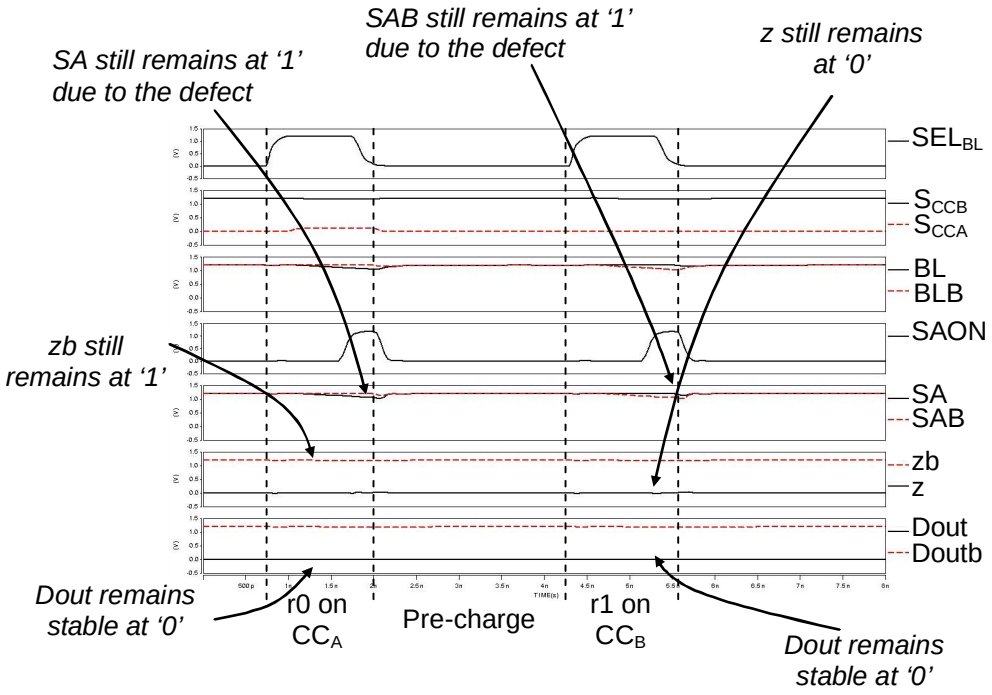

Figure I.25 – Waveforms of $< 0r0, 1r1/1/c >$ simulation (Df3)

This simulation starts on two different core-cells (CC_A and CC_B) belonging to the same group of columns (*i.e.* sharing the same sense amplifier) with CC_A containing a logic '0', CC_B a logic '1' and Data_out initialized at a logic '0'.

A $r0$ operation is first applied on CC_A. BL node is discharged and BLB node remains at Vdd. Then, the SAON signal is activated to enable the sense amplifier. However, due to the presence of the defect, it remains disabled, *i.e.* zb remains at logic '1' and z at logic '0' instead of z = zb = 0 (see Table I.3 for a $r0$ operation). The data output circuitry is in a memory state and thus does not change. It remains at logic '0'. The fault is not observed as the read data (a logic '0' in our case) is the same than that initially stored in the data output circuitry (a logic '0').

Then, a second read operation is performed with a $r1$ on CC_B. BL node remains at Vdd and BLB node is discharged. Once again, both nodes SA and SAB remain at logic '1' due to the defect, thus implying z = 0 and zb = 1 instead of z = zb = 1. The data output circuitry is in a memory state, implying that it still provides a logic '0' instead of a logic '1'. The fault is therefore sensitized and observed during the second read operation.

Note that if node Data_out is known to be initially at a logic '1', only one read operation is necessary to observe the fault in this case. However, in order to cover all possible cases, we must apply two read operations with opposite data on the same sense amplifier to be sure to detect a d2cIRF1 as the initial Data_out value is unknown.

I.3.3.3. March test solution

As shown previously, a d2cIRF1 may occur in presence of defect Df3. Such a faulty behavior is sensitized and observed with a specific sequence of read operations. This sequence is defined as follows:

$$rx\ r\bar{x}$$

where both read operations have to be obviously performed on two distinct core-cells sharing the same sense amplifier.

We formulate below some remarks about the possible modifications allowed on this sensitized operation sequence:

Remark 1: *An important property is that when a rx operation is performed by a sense amplifier, the Data_out node of the corresponding data output circuitry remains stable as long as a $r\bar{x}$ operation is not performed by a sense amplifier that shares the same data output circuitry. Consequently, any type of write operation in the memory may be allowed between these two read operations.*

Remark 2: *Obviously, several rx operations through all sense amplifiers sharing or not the same data output circuitry do not change the Data_out node value. Consequently, it may be allowed to perform any number of rx operations between the $rx\ r\bar{x}$ operations all over the memory.*

Remark 3: *If a $r\bar{x}$ operation is performed with another sense amplifier that does not share the same data output circuitry than the targeted one, then the Data_out node driven by the targeted sense amplifier is not disturbed. Consequently, any $r\bar{x}$ operation may be performed with all other sense amplifiers that do not share the targeted data output circuitry.*

These different remarks allow a less stringent sequence of sensitization for the d2cIRF1 detection as presented in Figure I.26.

From this statement, it is easy to create a specific March test to detect d2cIRF1s. However, as previously seen for SWDF testing, it is more interesting to obtain a March test that covers a larger set of fault models rather than only d2cIRF1s. We have thus to look for possibilities to embed or find the required successive operations for d2cIRF1 detection in existing March algorithms. We propose here to analyze if the March C- algorithm is able to detect d2cIRF1. For more simplicity, let us just redefine this algorithm in Figure I.27

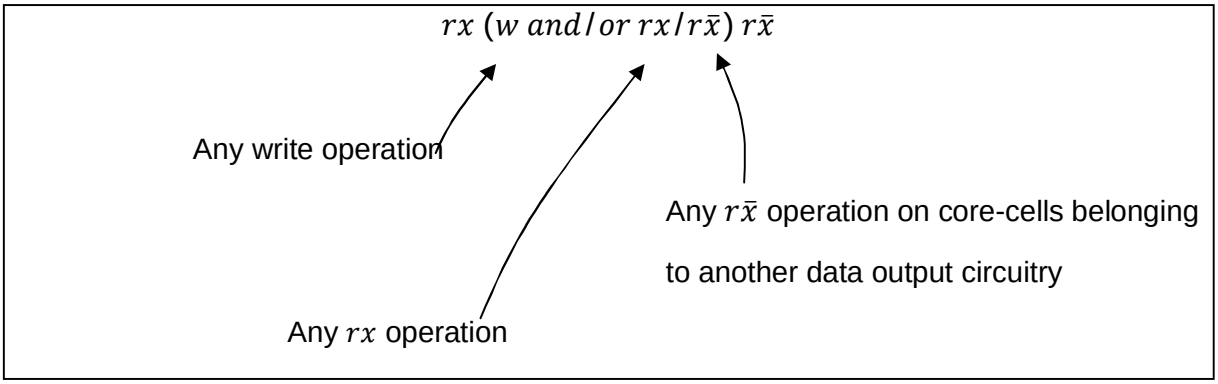

Figure I.26 – Relaxed constraints to detect d2cIRF1

In the March C-, the successive March elements M1/M2, M2/M3, M3/M4 and also M4/M5 feature the required sensitization sequence ($r0r1$ or $r1r0$) but they do not allow the detection of d2cIRF1 in all sense amplifiers. Let us consider a memory structure in which four sense amplifiers share the same data output circuitry. Whatever the addressing order, March element M1 performs a $r0$ operation on all the core-cells of the memory, meaning that all data output circuitries are set to a logic '0'. During this element, $w1$ operations are also performed but have no influence on data output circuitries (*c.f.* Remark 1).

$$\underset{M0}{\updownarrow (w0)} \underset{M1}{\uparrow (r0, w1)} \underset{M2}{\uparrow (r1, w0)} \underset{M3}{\downarrow (r0, w1)} \underset{M4}{\downarrow (r1, w0)} \underset{M5}{\updownarrow (r0)}$$

Figure I.27 – March C- algorithm

Then, March element M2 is applied using the same addressing order as M1. The first targeted core-cell is selected for a $r1$ operation. If the sense amplifier corresponding to this core-cell is affected by Df3, a logic '0' is read (this is the logic data previously stored in the corresponding data output circuitry) instead of a logic '1'. The fault is therefore sensitized and observed. Otherwise, if this first sense amplifier works correctly, the read data is a logic '1' and then the corresponding data output circuitry stores a logic '1'. According to Remark 3, it is then impossible to detect the fault in the three other sense amplifiers sharing this data output circuitry. With the application of March elements M1/M2 we can only detect a d2cIRF1 affecting the first selected sense amplifier among a group of four sense amplifiers sharing the same data output circuitry (using the $\uparrow$ addressing order). In the same way, the application of March elements M3/M4 allows the detection of d2cIRF1s affecting the first sense amplifier among a group of four sense amplifiers sharing the same data output circuitry (using the $\downarrow$ addressing order).

At this point, a straightforward solution should consist in applying the two read operations ($rx\ r\bar{x}$) in a March element. The proposed solution consists in using the modifications of the March C- presented in [DIL04a]. In this paper, the authors have proposed a new March test called March iC- (Figure I.28) for ADOFs (Address Decoder Open Faults) detection. The particularity of this new March is that it performs each read/write operation with an alternated data value Av where v is the initial value. In addition, it uses a specific addressing order (with an hamming distance of one between two consecutive addresses). It is also important to notice that these modifications (data and addressing order) are allowed by DOFs of March test and hence do not change the fault coverage of the former targeted faults. It means that the March iC- still detects the fault models formally detected by the March C-.

$$\updownarrow (wAv) \uparrow (rAv, wA\bar{v}) \uparrow (rA\bar{v}, wAv) \downarrow (rA\bar{v}, wAv) \downarrow (rAv, wA\bar{v}) \updownarrow (rA\bar{v})$$

Figure I.28 – March iC- algorithm

Using the concept of alternated data of the March iC-, we have now to find the good addressing order to guarantee the detection of all d2cIRF1s. Let us consider element M1 and v = 0. The successive operations applied at different addresses are:

$$(r0, w1), (r1, w0), (r0, w1), (r1, w0) \ \ldots$$

$$Add1 \quad Add2 \quad Add3 \quad Add4 \ \ldots$$

At this point, there are many possibilities to obtain the sequence of sensitization. But the simplest solution is to address with Add1 a core-cell that uses a sense amplifier and with Add2 another core-cell that uses the same sense amplifier. Consequently, we perform $r0$, $r1$ operations with a $w1$ between them that does not disturb the detection (*c.f.* Remark 1). Among the possible addressing orders, the simplest ones are the column after column or the line after line addressing orders. Let us first consider the column after column addressing order and the memory structure presented in Figure I.21. CC_{00} is selected for a $r0$ and a $w1$ operations. Then CC_{01} (the core-cell on the next line) is selected for the $r1$ and $w0$ operations. The fault is therefore sensitized and observed by the couple ($r0$, $r1$). In the same way, with the line after line addressing order, the first targeted core-cell is CC_{00} and the second is CC_{10} (the core-cell on the next column) in which we perform $r0$ and $r1$ operations respectively.

I.3.4. d2cIRF2 analysis

In this section, we detail the behavior of the sense amplifier affected by a d2cIRF2. As previously done, we first provide a FFM of the faulty behavior by using FPs. Next, we present electrical measurements to analyze the impact of a d2cIRF2 on the SRAM behavior. Finally, we propose a possible March test solution to detect d2cIRF2s.

I.3.4.1. Functional fault modeling

In presence of defects Df4 to Df9 a d2cIRF2 may occur. From these defects two groups can be constructed:

- Group 1: Df4, Df7 and Df9 are defects impacting the pull up of z and zb outputs.
- Group 2: Df5, Df6 and Df8 are defects impacting the pull down of z and zb outputs.

Let us first analyze defects of group 1. As these defects prevent the pull up of z and zb, they impact the $r1$ operation (see Table I.3). To sensitize defects of group 1 we must first set nodes z and zb to a logic '0'. This configuration corresponds to a $r0$ operation (see Table I.3). Consequently, detection of defects belonging to group 1 requires a $r0$ operation to initialize z and zb nodes at logic '0', followed by a $r1$ operation for the sensitization.

In the same way, as defects belonging to group 2 prevent the pull down of z and zb, they impact the $r0$ operation. Consequently, detecting these defects requires a $r1$ operation to initialize z and zb nodes at logic '1', followed by a $r0$ operation for the sensitization.

Based on these descriptions, a d2cIRF2 can be defined with two FPs as follow:

FP1: $< 0\boldsymbol{r0}, \boldsymbol{1r1/1/0} >$ A $r0$ is performed on a first core-cell. Then, a $r1$ is performed in another core-cell sharing the same sense amplifier. A logic '0' is read on node Data_out instead of a logic '1'. This FP is related to defects of group 1.

FP2: $< 1\boldsymbol{r1}, \boldsymbol{0r0/0/1} >$ A $r1$ is performed on a first core-cell. Then, a $r0$ is performed in another core-cell sharing the same sense amplifier. A logic '1' is read on node Data_out instead of a logic '0'. This FP is related to defects of group 2.

Note that we do not provide electrical simulations for each defect implying a d2cIRF2 as Df4, Df7 and Df9 induce the same faulty behavior, and faulty behavior in presence of Df5, Df6 and Df8 can be obtain by duality. Consequently, the next section is only dedicated to an electrical study in presence of Df4.

I.3.4.2. Electrical simulations with Df4

Waveforms in Figure I.29 present the faulty behavior of the memory in presence of Df4. They were obtained with typical PVT conditions (typical process, 1.2V supply voltage, 27°C) and a defect size of about 500 kΩ. This simulation involves two different core-cells (CC_A and CC_B) belonging to the same group of columns (*i.e.* sharing the same sense amplifier) with CC_A containing a logic '0', CC_B a logic '1' and Data_out initialized at logic '1'.

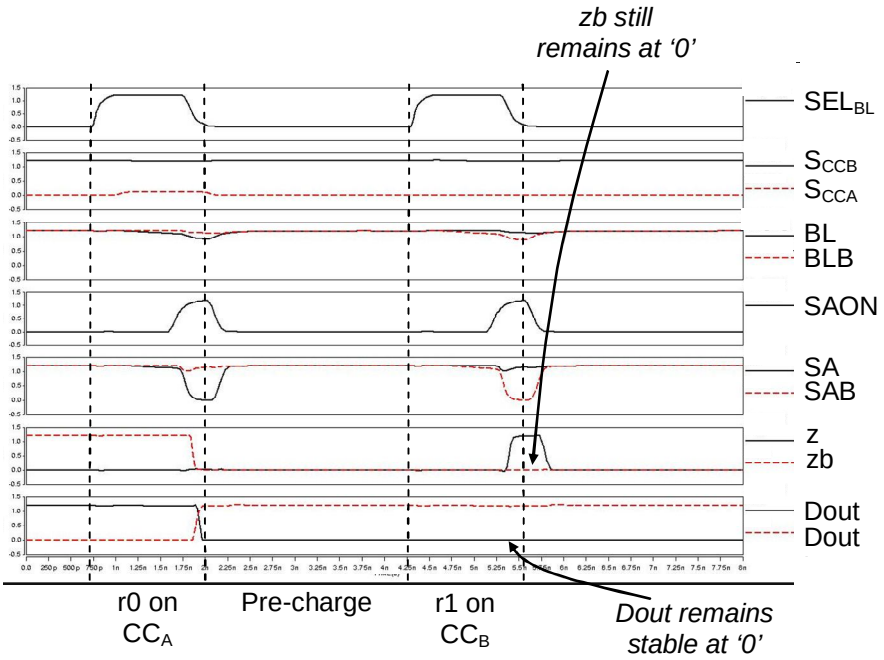

Figure I.29 – Waveforms of $< 0r0, 1r1/1/0 >$ simulation (Df4)

A $r0$ operation is first applied on CC_A. BL node is slightly discharged (about 100mV) and BLB node remains at Vdd. Then, the SAON signal is activated to enable the sense amplifier. This first operation is correctly acted as zb is correctly pulled down. At the end of this $r0$ operation, node Data_out presents a logic '0'.

Then, pre-charge circuits are switched on. All the lines (BL, BLB, SA and SAB) are therefore forced to Vdd, normally implying node z to be set at logic '0' and node zb to be set at logic '1'. However, due to the presence of Df4, zb remains at logic '0'.

A second read operation is then performed with a $r1$ on CC_B. BL node remains at Vdd and BLB node is discharged. Then, the sense amplifier is enabled by its SAON signal. SA remains at Vdd whereas SAB is fully discharged. Thus, node z flips to a logic '1'. However, due to Df4, node zb still remains at logic '0'. The data output circuitry is then in a memory state (*c.f.* Table I.3). Data_out still provides a logic '0' instead of a logic '1'.

I.3.4.3. March test solution

As previously shown, a d2cIRF2 may occur in presence of defects Df4 to Df9. Such a faulty behavior are sensitized and observed with specific sequences of read operations. These sequences are defined as follows:

- $r0\ r1$ for defects belonging to group 1
- $r1\ r0$ for defects belonging to group 2

where both operations have to be performed on two distinct core-cells sharing the same sense amplifier.

As previously done for d2cIRF1 we can try to find less stringent detection sequences, *i.e.* allow additional read or write operations between the two read operations require for d2cIFR2 detection. Nevertheless, as defects impact pull up or pull down of z and zb nodes, any read or write operations may mask the fault effect.

For a complete understanding, we have simulated the memory functioning in presence of Df4. Waveforms in Figure I.30 were obtained for worst case conditions (process: slow, voltage: 1.08V, temperature: -30°) with Df4 = 140kΩ. As shown in Table I.4, with these conditions the memory is affected by a d2cIRF2 when we perform a $r0$ immediately followed by a $r1$ operation. To confirm the fact two read operations must be applied sequentially, we have simulated the memory functioning by applying the following sequence of operations:

$$r0 \text{ on } CC_A,\ w1 \text{ on } CC_B \text{ and } r1 \text{ on } CC_B$$

where CC_A and CC_B are two core-cells sharing the same sense amplifier and containing a logic '0'.

Let us now detail the simulations presented in Figure I.30. First a $r0$ operation is applied on CC_A. BL node is discharged and BLB node remains at Vdd. Then, the SAON signal is activated to enable the sense amplifier. This first operation is correctly acted as zb is correctly pulled down. Node Dout provides a logic '0'.

Then, pre-charge circuits are switched on. All the lines (BL, BLB, SA and SAB) are therefore forced to Vdd, implying z to be set at logic '0' and zb to be set at logic '1'. However, due to the presence of the defect the inter node (see Figure I.22) is not correctly pull down. Consequently, zb remains at logic '0'.

Then, a write operation is performed on the second core-cell CC_B. This operation is correctly acted. However, during this time, node inter is enough discharged and reaches the threshold voltage of $Vdd/2$ implying that zb flips to logic '1'. Consequently, the fault effect is masked.

Finally, a second read operation is applied on CC_B which contains a logic '1'. The faulty behavior of the sense amplifier is masked as node zb has reaches Vdd before the read operation begins.

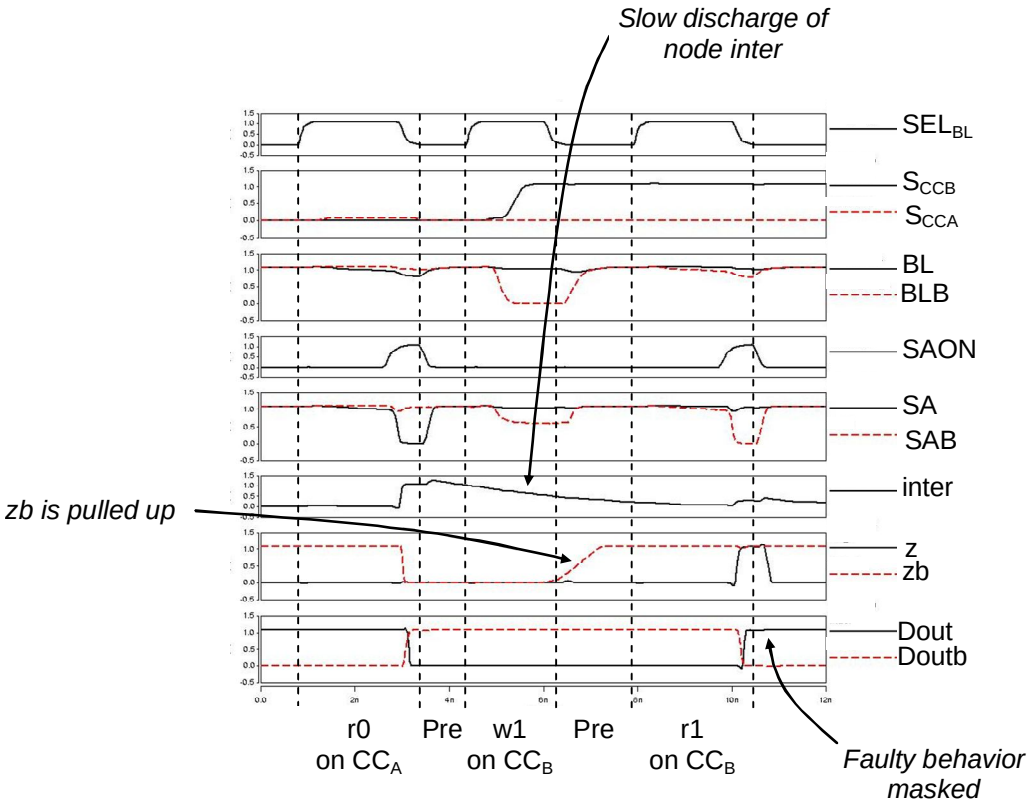

Figure I.30 – Waveforms of $< 0r0, 0w1r1 >$ simulation (Df4)

Consequently, we have to find a March algorithm which contains two successive read operation with opposite data value. The March iC- algorithm described in sub-section I.3.3.3 is able to detect such faulty behavior.

In fact, if we consider element M5 (see Figure I.28), the succession of operation applied at different addresses is:

$$(r0) \quad (r1) \quad (r0) \quad (r1) \quad \dots$$
$$Add1 \quad Add2 \quad Add3 \quad Add4 \quad \dots$$

Two successive read operations have to be applied on the same sense amplifier. The simplest way to do that is also the line after line or the column after column addressing order. Let us first consider the column after column addressing order and the memory structure presented in Figure I.21. CC_{00} is selected for a $r0$ operation. Then CC_{01} (the core-cell on the next line) is selected for the $r1$ operation. The fault is therefore sensitized and observed by the couple ($r0$, $r1$). In the same way, with the line after line addressing order, the first targeted core-cell is CC_{00} and the second is CC_{10} (the core-cell on the next column) in which we perform $r0$ and $r1$ operations respectively.

Based on these statements, we can say that the March C- algorithm with a specific data (alternated data value) and a specific addressing order (line after line or column after column) is a suitable solution to detect all d2cIRF2 that may affect sense amplifiers of an SRAM. Others solutions can also be found, especially for the addressing order, but are less conventional compare to the line after line or column after column addressing orders.

I.3.5. Conclusions

In this chapter, we have analyzed and characterized the effects of resistive-open defects that may occur in the sense amplifiers of SRAMs. We have shown that several resistive-open defects may lead to new types of dynamic behaviors which have never been experienced in the past. These faulty behaviors have been modeled as a d2cIRF1 and d2cIRF2. There are two distinct ways to qualify this behavior:

- d2cIRF1: all read operations cannot be acted.
- d2cIRF2: only $r0$ or $r1$ operation cannot be acted depending on the defect location.

Such fault models are a consequence of failures in the sense amplifier which prevent it to perform any read operations (in case of type 1) or only a single type of read operation (either $r0$ or $r1$ in case of type 2). We have performed electrical simulations to give a complete understanding of such faulty behaviors.

The conclusion of this study is that the March iC- algorithm with a particular addressing order (line after line or column after column) is able to detect all types of d2cIRFs.Table I.5 summarizes the ability of March iC- elements to detect d2cIRF1 and d2cIRF2, assuming that the core-cell contents are initialized by a previous write. It is also important to notice that these modifications do not change the ability of March iC- to detect the former targeted faults (stuck-at, transition, coupling etc …).

	d2cIRF1	d2cIRF2
March iC- element	M1 to M5	M5

Table I.5 – March iC- ability

Chapter 4. Influence of threshold voltage deviations in SRAM core-cells

Until recently, failure mechanisms were fairly simple. One gate was subject to a "hard fault". For example, a speck of dust felt on a track causing a resistive-open or a short.

Nowadays, as the silicon industry moves towards the end of the technology roadmap, controlling the manufacturing of scaled devices is becoming a great challenge. In VDSM technology, global (inter-die) and local (intra-die) device parameter variations are expected to be more and more significant [BOR03a]. These fluctuations are more pronounced in minimum geometry transistors commonly used in area-constrained circuits such as memories, especially core-cells which break layout rules.

A wafer may be subject to global variations; a gradient of dopant concentration may be observed. In this case, all transistors are subject to the same kind of parametric deviation. On the other hand, local variations, resulting from mismatches in parameters of similar transistors (threshold voltage – V_{TH}, geometry – L/W, mobility, etc), are as large as transistors use minimum geometry. These mismatches modify the strength of individual transistors and thus may lead to new types of failure in memories.

Among the possible sources of deviation, also called mismatch, the intrinsic fluctuation of V_{TH}, which is the main source of deviation due to random dopant effect [BHA01], has been studied in [BOR03a]. In this study, the authors present a qualitative analysis of V_{TH} mismatch impacts. They show that V_{TH} mismatches in an SRAM core-cell may induce a read or write failure. This study does not provide manufacturing data on possible location of V_{TH} mismatch in the core-cell. Moreover, there is no simulation result with different values of V_{TH} mismatches, and no analysis on PVT (Process, Voltage, Temperature) conditions. Nevertheless, this study is of importance as it pinpoints new problems and opens new ways for nanoscaled SRAM testing.

In this chapter, we consider threshold voltage (V_{TH}) variations in SRAM core-cells. For internal reasons, these studies are done on a memory designed with Infineon 90nm technology. We first provide an analysis of read and write operations to determine which transistor of the core-cell will have an impact on the memory function if it is mismatched. Then, a mismatch injection is performed and results show that the behavior of the core-cell is impacted with more or less complex failure mechanisms. Identified fault models related to the considered V_{TH} mismatches are Transition Faults (TF), Read Destructive Faults (RDF)

[VDG00] and dynamic Read Destructive Faults (dRDF) [ADA96, HAM02]. We show that the process (P) and temperature (T) have a large impact on the resulting faulty behaviors due to the V_{TH} mismatch injection.

The rest of the chapter is organized as follows. Section 1 presents the simulation flow used for mismatch injection. Section 2 provides an analysis of read and write operations to determine which transistors of the core-cell are candidates for V_{TH} mismatch injection. Section 3 presents the simulation results obtained and gives the test requirements for an effective mismatch detection. Finally, Section 4 concludes the chapter.

I.4.1. Simulation flow

In presence of parametric deviations, the characteristics of two neighbor transistors may significantly change, following statistical distribution laws. Such deviations are called local variations or transistor mismatches. Transistor currents are impacted by those fluctuations. The following equation gives the classical simplified MOS current:

$$I_{ds} = \frac{1}{2} \times k \times \frac{W}{L} \times (V_{gs} - V_{TH})^2 \qquad \textit{(Eq. I.2)}$$

where:

$$V_{TH} = V_{TH0} + K_{be} \times \left(\sqrt{|V_{sb}| + 2\Phi_F} - \sqrt{2\Phi_F} \right)$$

$$k = \mu \times C_{ox}$$

The transistor drain-source current (I_{ds}) is proportional to the mobility (k) and also depends on the threshold voltage (V_{TH}). Mobility mismatches affect I_{ds} slope whereas threshold voltage mismatches change the curve threshold, *i.e.* the higher the threshold voltage, the lower the current.

In this study we consider only threshold voltage mismatches as they are the main sources of deviation due to random dopant effect [BHA01]. This parameter follows a Gaussian distribution and a maximum of 6σ deviation (six times the standard deviation) is generally considered in VDSM technologies.

The impact of V_{TH} mismatches has been simulated with the following varying parameters:

- Process corner: slow, typical, fast, fast n / slow p, slow n / fast p
- Supply voltage: 0.9V, 1.2V, 1.5V
- Temperature: -40°C, 27°C, 125°C

V_{TH} mismatch varies from 0σ up to |6σ|. The same variations were added either to one single transistor or to a combination of transistors enabling a comparison between these situations.

No Monte-Carlo simulations were run. The method applied in this study consists in injecting mismatches to most sensitive transistors of the core-cell. Candidate transistors for mismatch injection on the core-cell are extracted from the analysis of read and write operations presented in the next Section.

I.4.2. Mismatch sensitivity during read/write operations

V_{TH} mismatches may affect all transistors of a core-cell but, according to the performed operation (read or write), only some of them are important. In order to determine which transistor is candidate for V_{TH} mismatch injection, we present in this Section a complete analysis of write and read operations.

For write operations, only V_{TH} mismatches that reduce the core-cell transistor conductivity are considered. Let us consider the core-cell presented in Figure I.31 in which the cell originally stores a logic '1'. Node S is at Vdd and node SB at Gnd. Remember that to write a logic '0' ($w0$) into this core-cell, BLB line remains at Vdd, BL line is lowered to Gnd and the cell is selected by applying Vdd on WL. Operating devices and current flows during this $w0$ operation are illustrated in Figure I.31. A current flows from S to BL through *Mtn3*, discharging C_s. As the voltage at node S decreases, *Mtp1* starts to conduct. In the same way, C_{SB} is charged by the current flowing through *Mtn4*. The voltage at node SB increases, involving the conduction of *Mtn2*. This write analysis shows that four transistors (*Mtn3*, *Mtn4*, *Mtp1* and *Mtn2*) are involved during the $w0$ operation. We can easily verify that *Mtp2* and *Mtn1* in addition to pass transistors are used for a $w1$ operation.

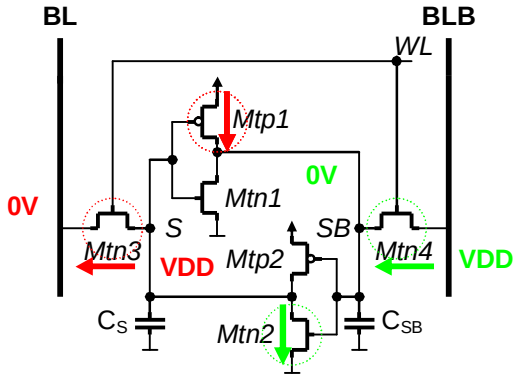

Figure I.31 – Core-cell currents whose weakness is critical for a $w0$ operation

Waveforms of the different currents and voltage levels induced by the $w0$ operation are reported in Figure I.32. These curves show that the voltage at node S reaches $Vdd/2$ before node SB. Thus, node S is controlling the $w0$ operation. Conversely, node SB will control the $w1$ operation. From this, we can say that V_{TH} mismatches will have an impact during a $w0$ operation if they affect *Mtn3* and/or *Mtp1* transistors (respectively *Mtn4* and/or *Mtp2* transistors for a $w1$).

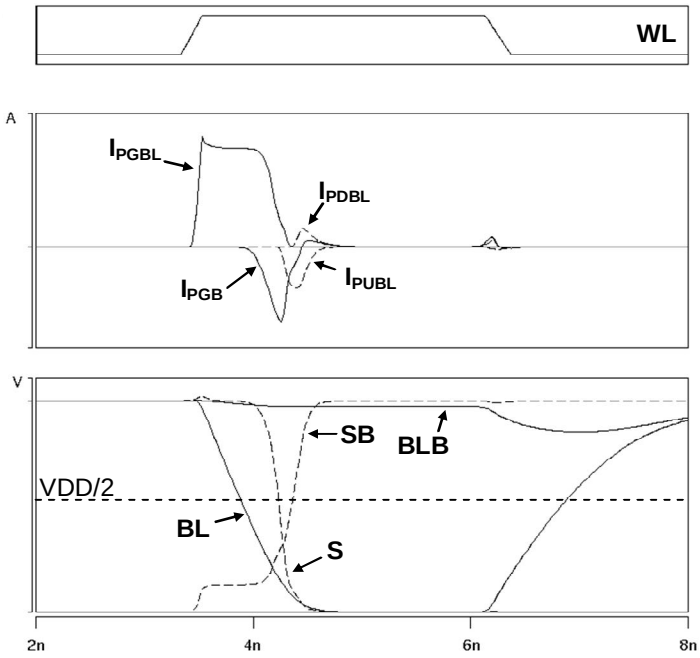

Figure I.32 – Currents and voltages during a $w0$ operation

In the same way, we analyze which transistors of the core-cell are involved during a read operation. In this case, only transistors that influence the total current discharging the bit line, but also the core-cell stability (ability to keep the stored data) are considered.

Let us assume that the cell has stored a logic '0'. Operating devices and current flows during this read operation are illustrated in Figure I.33. In this case, node S is at Gnd and node SB is at Vdd. Before the read operation, BL and BLB lines are pre-charged at Vdd. When the word line is selected (WL signal being high), the two pass transistors *Mtn3* and *Mtn4* are turned on and the pre-charge circuit is turned off, implying a Vdd floating level on BL and BLB. As the potential of node SB and BLB are the same, no current flows and transistors *Mtp1* and *Mtn4* will maintain the Vdd level at node SB. On the other side of the core-cell, a current flows from BL through transistors *Mtn3* and *Mtn2*, thus discharging the equivalent capacitance C_{BL} of the bit line initially charged at Vdd.

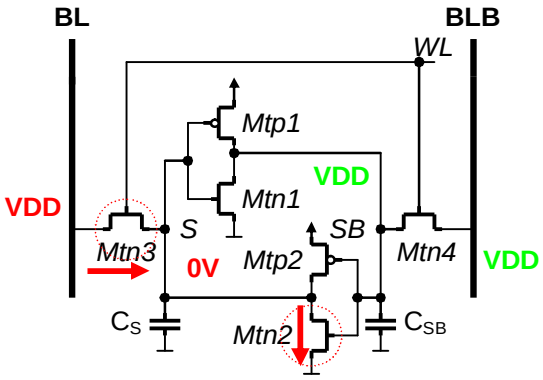

Figure I.33 – Core-cell currents whose weakness is critical for a $r0$ operation

Waveforms of currents and voltages involved during a $r0$ operation are presented in Figure I.34. At the end of the $r0$ operation, node BL is discharged. The differential voltage between BL and BLB nodes (ΔBL), is measured by the sense amplifier to provide a logic data output. In this case, ΔBL is negative and thus the sense amplifier will provide a logic '0'.

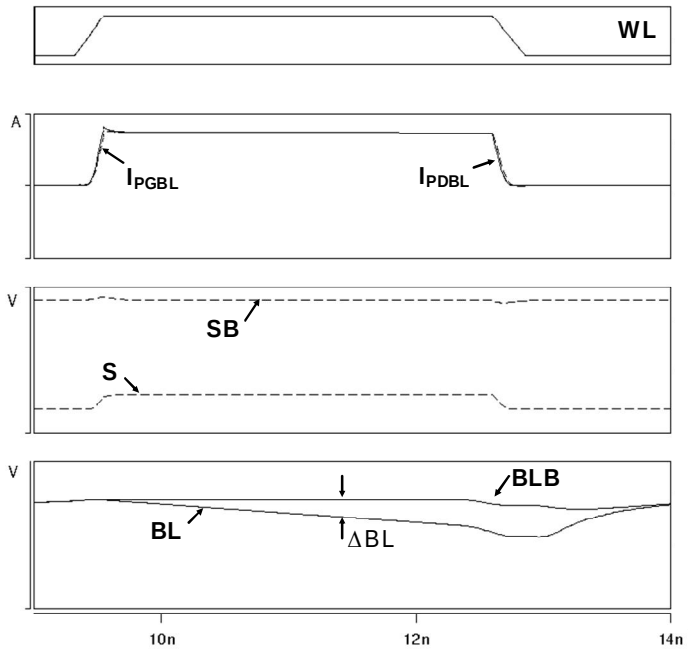

Figure I.34 – Currents and voltages during a $r0$ operation

This analysis demonstrates that V_{TH} mismatches on *Mtn3* and/or *Mtn2* transistors will have an impact on the $r0$ operation (*Mtn4* and/or *Mtn1* transistors for a $r1$). In the next Section, we show experimental data demonstrating the impact of V_{TH} mismatches on the transistors.

I.4.3. Mismatch related fault models

The previous section has described write and read operation mechanisms. They are quite complex, involving transistors of the core-cell which differ depending on the operation and the data stored in the core-cell. From these analyses, we have performed a mismatch injection in different locations of the core-cell as presented in Figure I.35. The goal here is to provide a functional fault modeling of each mismatch configuration.

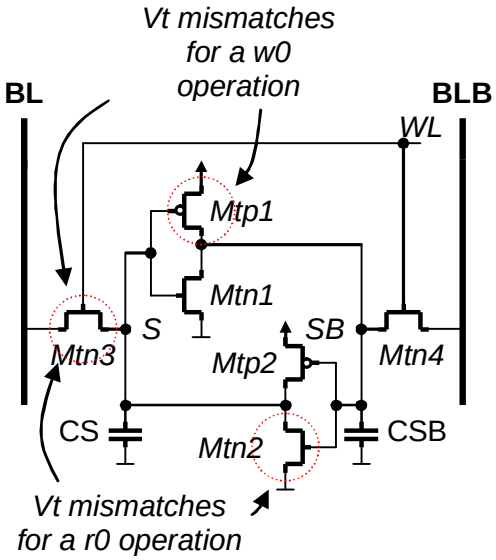

Figure I.35 – Considered V_{TH} mismatch locations for $w0$ and $r0$ operations

I.4.3.2. Result overview

Simulations were performed considering single or double mismatch locations with identical V_{TH} deviations (up to 6σ). Moreover, these simulations were done under the most constraining PVT conditions to extract the one that maximize the fault detection (*i.e.* the minimum detected V_{TH} mismatch). Results are reported in Table I.6.

Mismatch location	Fault Model	Mismatch size	PVT
Mtn3	TF	~ 4σ	sf, 0.9V, -40°C
Mtn3 & *Mtp1*	TF	~ 4σ	sf, 0.9V, -40°C
Mtn3	RDF	~ 6σ	fs, 0.9V, 125°C
Mtn3 & *Mtn2*	RDF	~ 3σ	fs, 0.9V, 125°C
Mtn3	dRDF	~ 3.8σ	sf, 0.9V, -40°C

Table I.6 – Results summary

The first column gives the location of the V_{TH} mismatch (see Figure I.35) and the second one indicates the type of fault model observed. The third column gives the minimum mismatch value that sensitizes the fault and the last column gives the PVT conditions that maximize the mismatch detection (*i.e.* worst case conditions).

The first result of these simulations is that PVT conditions that maximize the mismatch detection are always at low voltage (0.9V). In fact, a V_{TH} variation of 100mV is proportionally higher for a supply voltage of 0.9V than for a supply voltage of 1.5V (see *Eq. I.2*). This first result shows that V_{TH} deviations have their main impact at low voltage while hard defects, such as resistive-open defects in the core-cell, better manifest themselves at high voltage [BOR03b].

As a second result on PVT conditions, it is important to notice that temperature corners are the extreme ones (-40°C and +125°C). This phenomenon is explain by the fact that V_{TH} varies in a monotonously way with the temperature (linear relationship), it means V_{TH} is strictly decreasing when the temperature increasing (see *Eq. I.3*). Thus, the extreme corners maximize the detection of mismatches.

$$V_{TH}(T) = V_{THnom} + CTE \times (T_{nom} - T) \qquad \textit{(Eq. I.3)}$$

$$\Longrightarrow \frac{dV_{TH}(T)}{dT} = -CTE \leq 0$$

For a test applied at room temperature (+27°C for example) the same faulty behaviors can be obtained but associated with higher mismatch values.

Faults observed are TF (already defined), Read Destructive Faults - RDF (the cell loses its content during a read operation) and dynamic Read Destructive Faults - dRDF (the cell is not correctly written and loses its contents after one or several at-speed read operations). Each fault is induced by a different combination of mismatches, sensitizing sequences and PVT conditions.

As shown in Figure I.36 and Figure I.37, TFs occur when applying either a single or a combination of mismatches.

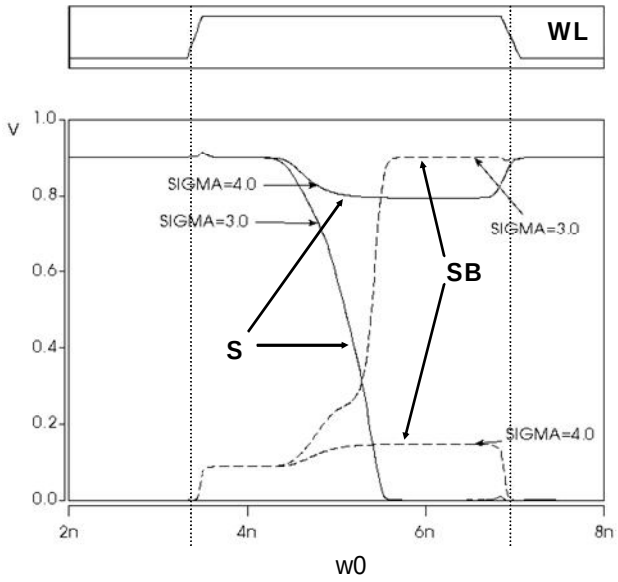

Figure I.36 – Transition Fault (sf, 0.9V, -40°C – *Mtn3*)

For these simulations, a $w0$ is applied on a core-cell that initially contains a logic '1'. For a V_{TH} mismatch higher than 4σ, a TF is observed in both cases, *i.e.* the write operation fails. Worst case conditions are, slow n / fast p, low voltage and low temperature.

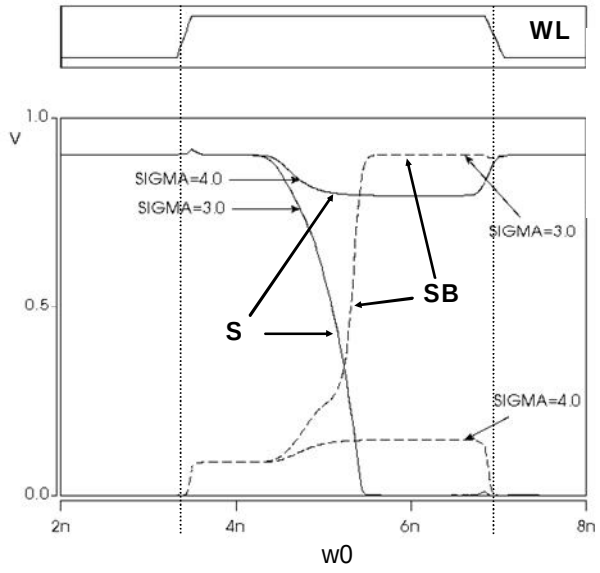

Figure I.37 – Transition Fault
(sf, 0.9V, -40°C – *Mtn3* & *Mtp1*)

RDFs are also observed for different combinations of V_{TH} mismatch. This time, a cell that initially contains a logic '1' is written to logic '0'. Then a read operation is performed. As can be seen in Figure I.38 and Figure I.39, the data is lost during the read operation when V_{TH} mismatch is higher than 6σ.

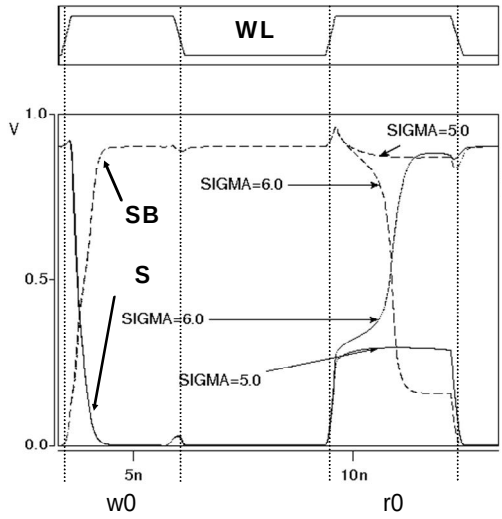

Figure I.38 – Read Destructive Fault
(fs, 0.9V, 125°C – *Mtn3*)

When a mismatch affects *Mtn3*, a RDF occurs only for a 6σ deviation. When two mismatches are considered on *Mtn2* and *Mtn3*, a RDF is clearly observed for a 3σ deviation. Worst case conditions are: fast n / slow p, low voltage and high temperature.

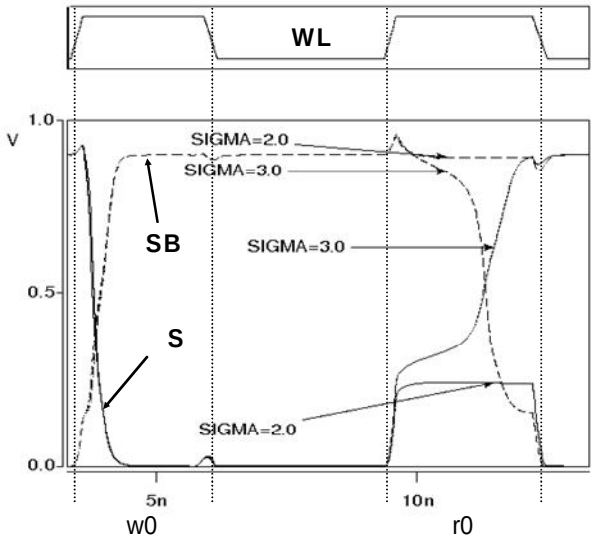

Figure I.39 – Read Destructive Fault
(fs, 0.9V, 125°C – *Mtn3* & *Mtn2*)

The last part of simulations performed shows that dynamic faults can also be observed, especially dRDF. To highlight such a behavior, we have first to discuss about the sensitizing sequence needed. A dRDF occurs when one or several read operations are performed at-speed on a core-cell just after a write operation on the same core-cell. Then, if the core-cell is defective, one of the read operations may induce a bit flipping in the core-cell. This faulty

behavior is due to a non complete write, *i.e.* the write operation does not allow nodes S and SB to stabilize at Gnd and Vdd respectively in case of a $w0$ operation. A read operation performed just after a non complete write makes the core-cell to possibly loose its content. So, both write and read operations are involved in the occurrence of a dRDF. The common transistor involved during these operations ($w0$ and $r0$ in our case) is *Mtn3*. In our study, this transistor is selected for a V_{TH} mismatch injection.

Figure I.40 shows a dRDF in which core-cell internal nodes (S and SB) are at an intermediate value at the end of the $w0$ operation. This defective core-cell looses its content during the second at-speed $r0$ operation for a 3.8σ deviation of V_{TH}. The worst case sensitization is the same as that found when TF occurs, *i.e.* slow n / fast p, 0.9V, low temperature.

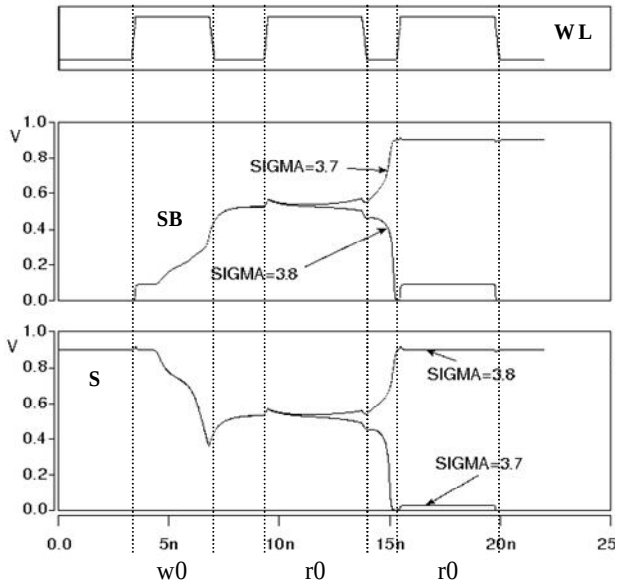

Figure I.40 – dynamic Read Destructive Fault
(sf, 0.9V, -40°C – *Mtn3*)

Further simulations have been performed with lower σ deviations of V_{TH}. However, in those cases, the flipping of the defective core-cell occurs after a higher number of successive read operations. This phenomenon is illustrated in Figure I.41. It shows that for a σ deviation higher than Border 2, a static fault is observed (a TF in our case). For lower σ deviation, dynamic faults occur (between Border 1 and Border 2). Positions of Border 1 and Border 2 depend on PVT conditions. In addition, position of Border 1 also depends on the number of read operations after the initial write operation. Finally, for σ deviations lower than Border 1, the core-cell operates properly.

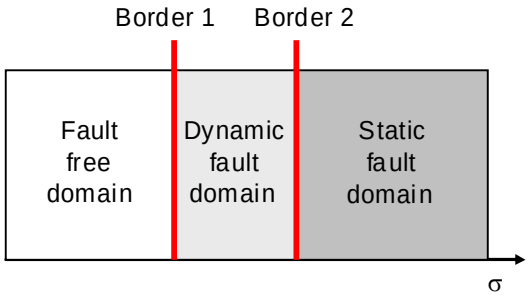

Figure I.41 – Fault type v.s. mismatch value

I.4.3.3. Test requirements

We have shown in the previous sub-section that a V_{TH} mismatch induces different faulty behaviors which can be modeled by TF RDF and dRDF. Now, we have to analyze the test requirements (algorithms and PVT conditions) needed to detect these fault models.

The selected test algorithm has to detect TF, RDF and dRDF. On one hand, the detection of TF is simple as most of the March algorithms have the ability to detect them. On the other hand, the detection of RDF and dRDF is more difficult as it requires a read (or multiple read) after a write operation. This succession of operations does not occur in classical March tests. Specific March, such as March RAW [ARS01] or March C- with specific addressing order [DIL04b], can be used. These two algorithms have also the ability to detect TF.

The problem is much more severe with respect to PVT conditions. First, V_{TH} mismatches have their main impact at low voltage while hard defects, such as resistive-open defects in the core-cell involving the same faulty behaviors, better manifest themselves at high voltage [BOR03b]. In addition, we have shown in the previous Section that, depending on the considered mismatch location, temperature and process have a large impact on the resulting fault model; process slow n fast p and low temperature for TF and dRDF, process fast n slow p and high temperature for RDF. These different PVT conditions make the test of SRAM core-cells more difficult. In fact, it is not possible to ensure the fault-free behavior of SRAM core-cells by applying a March algorithm in a unique PVT corner. This statement opens an additional problematic for the test of nanoscaled SRAMs.

I.4.4. Conclusion

In this chapter, we have analyzed and characterized the effects of V_{TH} mismatches that may occur in SRAM core-cells. We have first provided an analysis to determine which

transistors of the core-cell may have an impact during read and write operations of the memory if they are mismatched. Simulations performed with Infineon 90nm technology have shown that static (TF and RDF) and dynamic (dRDF) faults are obtained as resulting faulty behaviors of the V_{TH} mismatch injection. An important contribution of this study is also the analysis of PVT conditions for an effective test.

Conclusion

This part has been dedicated to an exhaustive study on resistive-open defects affecting SRAM write drivers and sense amplifiers as well as a study on local variations affecting the core-cell functioning.

Previous studies shown that resistive-open defects in core-cells, pre-charge circuits and address decoders lead to dynamic faults. In order to complete these studies, we have demonstrated that such defects in the write driver can also be the cause of dynamic behavior. Especially, some defects can cause a dynamic fault modeled as SWDF, some others lead to another dynamic faults modeled named URWF or URDWF (depending on the defect size). We also demonstrated that this kind of defect in the sense amplifier can also induce a faulty behavior called dynamic 2-cell Incorrect Read Fault (type 1 and type 2). Finally, March test targeting these dynamic faults have been developed. All these studies have been validated by electrical simulations performed with a 65nm CMOS Infineon technology.

Afterwards, we have analyzed and characterized the effects of V_{TH} mismatches impacting modules designed with minimum geometry transistors such as SRAM core-cells. We have first provided an analysis to determine which transistors of the core-cell may have an impact during read and write operations of the memory if they are mismatched. Simulations have shown that static (TF and RDF) and dynamic (dRDF) faults are obtained as resulting faulty behaviors of the V_{TH} mismatch injection. An important contribution of this study is also the analysis of PVT conditions for an effective test. Actually, the PVT conditions that maximize the mismatches detection are different from those that maximize the resistive-open defects detection. Consequently, the test of memory cannot be acted in a unique PVT corner. This study, realized in 90nm CMOS Infineon Technology (for internal reasons), opens the problem of mismatch influences in nanoscaled SRAMs. Further investigations have to be done in deeper technologies such as 65nm, 45nm, 32nm and 22nm for which the influence of parameter deviation should be much more severe.

Part II: Diagnostic of SRAMs

Introduction

Nowadays, the latest technologies present very high degree of integration allowing a number of circuits per die much higher than in the past. These new technologies are also more prone to defects, parasitic phenomena and manufacturing derives, which drastically reduce the yield. For this reason, fault detection, diagnosis and defect localization are used in order to repair defective memories thus improving SoC reliability and yield. In this part, we focus on diagnosis techniques dedicated to SRAMs.

Usually, techniques allowing memory repair identify the type of malfunction and try to find out its location as two separate phases [VDG98]. In order to reach good results in terms of repair, the information on the fault location is more important than the information on the nature of the fault itself. During memory diagnostics, a map of core-cells is made, with faulty and fault-free cells. On this base, particular algorithms optimize the use of spare columns and rows for the substitution of those containing the faulty cells. Conversely, when yield ramp up is targeted, the diagnosis approaches mainly focus on the identification of the cause of the failure as well as its location. In this way, layout and process optimizations are possible.

In this part, we consider two diagnosis methodologies. The first one is known as Design For Diagnosis (DFD) and targets only specific memory blocks (core-cell, pre-charge circuitry, write driver…). It consists in implementing extra hardware modules in the memory allowing to check given nodes or functionalities, *e.g.* bit lines voltage levels, core-cell strength... Such a technique suffers from an increase of the chip area. However, it provides essential and accurate information about faulty sites of the memory and is useful to enhance manufacturing process and/or design in the ramp up phase. In the literature, DFD modules are widely been developed to monitor core-cell functionalities. However, peripheral circuits have never been considered until now.

The second diagnosis approach does not target a specific block but instead takes a global approach to the problem and targets the detection of FFM. Existing diagnostic methods, based on a signature analysis [ABR90], generally resort to a fault dictionary and try to achieve the highest Diagnosability Ratio (DR) for a given test algorithm [CHA89, YAR96, NIG00, LI01]. *DR is defined as the ratio of the number of distinguishable fault types among the number of total detectable fault types.* However, signature-based diagnosis methods present two main drawbacks. First, as they use a fault dictionary, the possible fault models affecting the memory must be known before running the diagnosis procedure. Consequently, if a memory

is affected by a fault not considered in the fault dictionary, the diagnosis phase fails to provide any result or may provide a wrong response. Secondly, most of the existing signature-based solutions target only the diagnosis of static faults. Unfortunately, as seen in the first part of this thesis, dynamic faults become a major concern in recent SRAMs technologies.

This part is organized as follow. A first chapter presents two DFD modules able to deal with weak write drivers. The second chapter presents a new diagnosis approach that provides an alternative to signature-based approaches.

Chapter 1. Design For Diagnosis Solutions

This chapter presents two low cost DFD solutions for identifying weak or faulty write drivers. They consist in verifying logic and analog conditions that guarantee the fault-free behavior of the write driver. Both solutions allow a fast diagnosis (only three consecutive write operations are needed to fully diagnose the write driver) and induce low area overhead (about 0.5% for a 512x512 SRAM). Beside diagnosis, an additional interest of such solutions is their usefulness during a post-silicon characterization process, where they can be used to extract the main features of write drivers (logic and analog levels on bit lines).

This chapter is organized as follows: In the first Section, we expose a brief state-of-the-art before explaining a current-based DFD solution in Section 2. Section 3 is dedicated to a complete study on a voltage-based DFD solution. Finally, concluding remarks are provided in the fourth Section.

II.1.1. State-of-the-art

A DFD solution consists in implementing an additional hardware module able to point out specific memory functionalities. For example, cell stability is a major concern to evaluate the SRAM design reliability. It determines the sensitivity of the memory to process variations and operating conditions. So, monitoring such parameter presents a real relevance. Core-cells with lower cell stability than typical case are known as weak cells. Many works have been proposed in that way. These techniques are based on the fact that the state-restoring feedback (*i.e.* the inverter loop) of a weak cell is weaker or absent and thus they are more susceptible to write or read disturbs. All these techniques are divided in two categories, the single and programmable detection threshold techniques. The most known single threshold technique is called Weak Write Test Mode (WWTM) [MEI97]. Many implementation of such technique have been proposed [WEI01, SCH04]. In addition of a non regulate ability of threshold detection of such techniques, they present a non negligible extra area and some of them add extra design in the core-cell array. The programmable detections are described in [PAV04, PAV05, PAV06]. These techniques are based on the use of core-cells belonging to the same core-cell under test in order to act the stress.

Another DFD technique is provided in [PIL01] where the authors target the detection of strong resistive path through the path gates of core-cells. A targeted algorithm and a hardware module are designed to detect such faulty core-cells.

Many efforts have been done on DFD solution targeting core-cells functionalities. On the other hand, no works have been published on DFD solution for peripheral circuitry. Nevertheless, even if around 80% of the silicon area of a memory is taken by the core-cell array, which is hence more prone to defects than any other block, providing information on peripheral circuitry can save considerable amount of time during the ramp up phase in case of a malfunction coming from outside the core-cell array.

In the next section of this part, we propose two solutions providing information about write driver strength.

II.1.2. Requirements for fault-free operation of a write driver

The fault-free operation of the SRAM write driver has already been described in the first part of this thesis. Based on this description, we can enumerate the two important conditions that are needed to guarantee the fault-free behavior of the write driver – a logic and an analog conditions.

II.1.2.1. Logic condition

As shown previously, the write driver must act the pull down of one of the two bit lines. The other bit line is maintained at Vdd during the write operation. From this statement, we can extract a first condition for a fault-free operation of the write driver:

$$BL \oplus BLB = 1 \qquad \textit{(Eq. II.1)}$$

If this equation is not satisfied during a write operation, then it means that both bit lines present the same voltage level. In case of Vdd, no write operation is performed. Conversely, the two bit lines at Gnd indicate that both $w0$ and $w1$ operations are performed simultaneously.

This first condition allows performing a logical diagnosis of the write driver. Nevertheless, it does not allow verifying the exact voltage level driven on the bit lines during the write operation. Thus, an additional analog condition is needed to diagnose weak write drivers.

II.1.2.2. Analog condition

Voltage levels on bit lines during write operations are a major concern when embedded memories are used for high safety applications (automotive, medical…). In fact, over the lifetime of a product, memories are exposed to many phenomena (DC noise, coupling effects…) which degrade their performances. For this reason, it is important to verify the good voltage level of bit lines after manufacturing. A wrong level at this early stage of the lifetime of the memory indicates a weakness of the write driver, which can be degraded over the time and lead to erroneous write operations. So, in addition to the logic condition, an analog condition has to be satisfied to guarantee the good voltage levels on the bit lines.

The write driver can be seen as a current source that has to discharge one bit line and to maintain the other at Vdd. During a fault-free operation ($w0$) let us consider that it delivers a current I_{ideall} for the discharge of bit line BL and I_{idealh} for bit line BLB. Thus, a weak write driver delivers less current than I_{ideall} (resp. I_{idealh}). Consequently, at the end of the write operation, the level of the bit line that has to be discharged is higher than Gnd (resp. the level of the bit line that has to be maintained at Vdd is less than Vdd). This can be view on waveforms in Figure I.1 where a $w0$ operation is performed by a fault-free write driver (top of Figure I.1) and a weak write driver (bottom of Figure I.1).

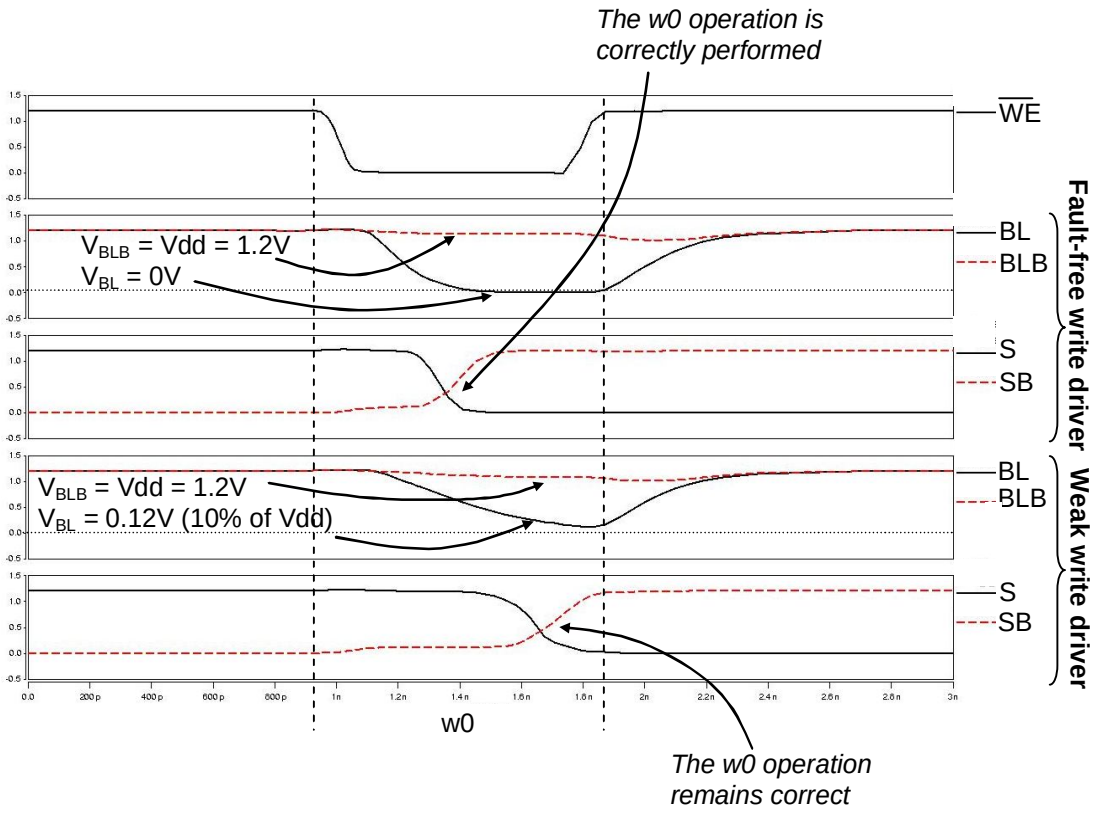

Figure II.1 – Fault-free and weak write driver operations

From this statement, we can extract two analog conditions for a fault-free operation in case of a $w0$ operation. Note that the analog conditions for a $w1$ operation can be derived in the same way.

$$I_{reall} \geq \alpha_1 \cdot I_{ideall} \text{ with } 0 \leq \alpha_1 \leq 1 \quad \textit{(Eq. II.2.a)}$$

$$\Rightarrow V_{BL} \leq \beta_1 \cdot Vdd \text{ with } 0 \leq \beta_1 \leq 1 \quad \textit{(Eq. II.2.a bis)}$$

and

$$I_{realh} \geq \alpha_2 \cdot I_{idealh} \text{ with } 0 \leq \alpha_2 \leq 1 \quad \textit{(Eq. II.2.b)}$$

$$\Rightarrow V_{BLB} \geq \beta_2 \cdot Vdd \text{ with } 0 \leq \beta_2 \leq 1 \quad \textit{(Eq. II.2.b bis)}$$

where α_1 and α_2 represent the strength of the write driver. Parameters β_1 and β_2 are derived from the α parameters and represent the level of charge and discharge of the bit lines. An ideal write driver will be defined by *Eq. II.2.a* and *Eq. II.2.b* with α_1 = 1 and α_2 = 1 implying $\beta_1 = 0$ (V_{BL} = 0V) and $\beta_2 = 1$ ($V_{BLB} = Vdd$).

Parameters α have to be selected depending on the memory technology and desired reliability level. In our case, we have considered a 65nm SRAM technology and we have chosen parameters as follows:

- α_1 insuring $V_{BL} \leq 0.1 \cdot Vdd$
- α_2 insuring $V_{BLB} \geq 0.7 \cdot Vdd$

Consequently, the write driver will be considered as faulty if it cannot discharged BL at a voltage lower than 10% of Vdd and maintain BLB at a voltage level higher than 70% of Vdd.

II.1.3. Description of the current-based DFD solution

The proposed DFD solution consists in adding a hardware module to verify both logic and analog conditions presented in the previous section. Note that we only present how to diagnose a weak or wrong $w0$ operation. The study of the $w1$ operation can be derived in a similar way.

II.1.3.1. Hardware diagnosis solution for the analog condition

The analog condition consists in verifying if the write driver delivers enough current in the bit lines. For a $w0$ operation, the bit line (BL) must be discharged at more than $\beta_1 \cdot Vdd$ by the current passing through transistor *Mtn1*. Respectively, BLB must be maintained at

$\beta_2 \cdot Vdd$ by the current passing through transistor *Mtp2*. A straightforward solution consists in sensing the resulting voltage levels on bit lines by using logic gates designed to have the required threshold voltage. However, such a solution is unpractical for two raisons:

- the difficulty to design gates with very low (0.1V) or very high threshold voltages
- the fact that we must sense two different voltages ($\beta_1 \cdot Vdd$ and $\beta_2 \cdot Vdd$) on each bit line to diagnose weak or wrong $w0$ and $w1$ operations.

Consequently, in order to use simple CMOS gates to sense bit line voltage levels, we propose to normalize the pass/fail diagnosis threshold voltage on bit lines at $Vdd/2$ (instead of 10% and 70% of Vdd). This is done by adding two transistors (*Mtptest* and *Mtntest*) producing a resistive divider bridge and hence modulating the bit line voltage levels. This principle is presented in Figure II.1.

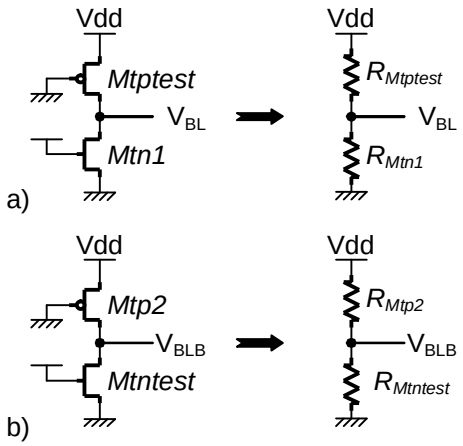

Figure II.1 – Principle of the DFD solution
a) for the low level and b) for the high level

In a stable state, transistors *Mtptest* and *Mtn1* (resp. *Mtntest* and *Mtp2*) can be seen as their equivalent resistances inducing the resistive divider bridge. The strength of *Mtptest* (resp. *Mtntest*) is chosen in order to have the following diagnosis conditions:

- if $V_{BL} < \frac{Vdd}{2} \Rightarrow$ the write driver satisfies the analog condition.
- if $V_{BL} > \frac{Vdd}{2} \Rightarrow$ the write driver does not satisfy the analog condition.

To be more precise on the sizing of transistors *Mtptest* and *Mtntest*, let us consider Figure II.2. It represents I_{ds} as a function of V_{ds} voltage levels of *Mtptest* and *Mtn1* transistors.

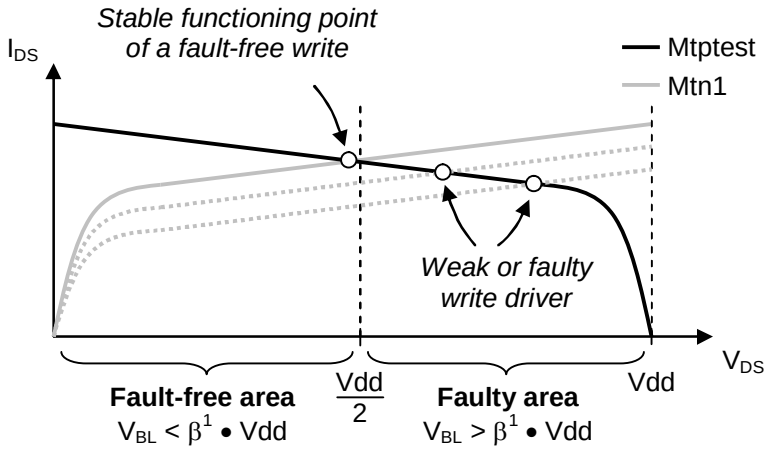

Figure II.2 – Principle of the diagnosis solution

The hardware implementation of such a principle is presented in Figure II.3. It is composed of two parts; the analog structure and the data processing providing the diagnosis result.

The analog structure embeds the two transistors *Mtptest* and *Mtntest* plus four transmission gates (*MtnpgBL*, *MtnpgBLB*, *MtppgBL* and *MtppgBLB*) and two inverters used to isolate and configure the diagnosis module. Two signals ($\overline{W0D}$ and $\overline{W1D}$ active at low level) control the configuration of the analog structure that depends on the write operation type ($w0$ or $w1$).

At the end of the write operation, the bit line level reflects the strength of the write driver. The analog structure is designed in order to obtain less than $Vdd/2$ on BL and more than $Vdd/2$ on BLB for a fault-free $w0$ operation. The data processing part allows translating these analog levels into a digital signal. Two inverters are used to amplify the signals and a XOR gate is used to provide the diagnosis results. Node S must be at logic ‘1’ during the write operation in case of a write driver satisfying the analog conditions.

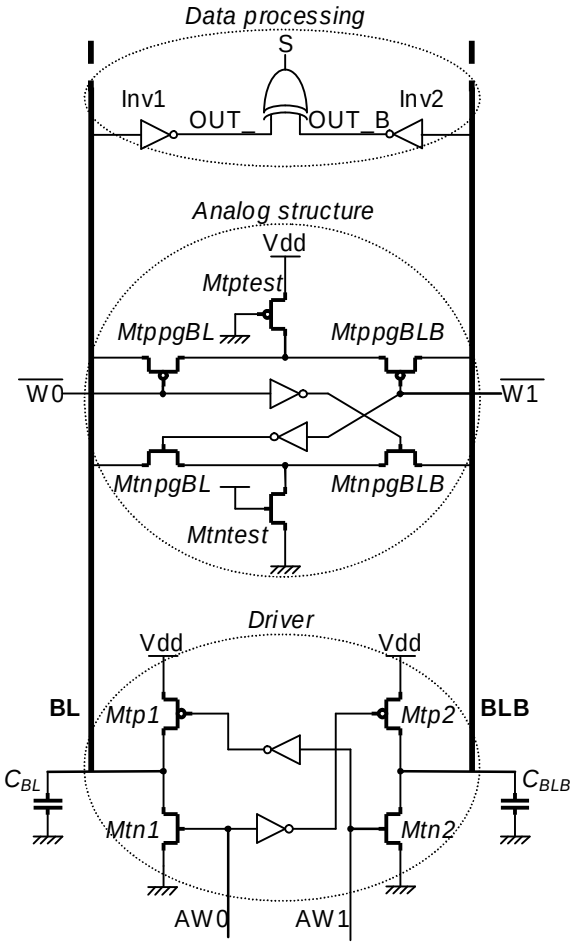

Figure II.3 – Hardware implementation of the diagnosis module

Waveforms in Figure II.4 illustrate the functioning of the proposed structure. Two simulations are superposed; a fault-free write driver simulation (continuous lines) and a weak write driver simulation (dotted lines).

At the beginning of the simulation, BL and BLB are pre-charged at Vdd. Then a $w0$ operation is performed, leading to AW0 = 1 and AW1 = 0. The diagnosis module is activated with $\overline{W0D}$ = 0 and $\overline{W1D}$ = 1. Then, BL node is discharged and reaches a level lower than $Vdd/2$ in case of a fault-free write driver. In case of a weak write driver, as transistor *Mtn1* has not enough strength to discharge the bit line, V_{BL} remains higher than $Vdd/2$. As diagnosis result, node S provides a logic '1' in case of a fault-free write driver and a logic '0' for a weak write driver.

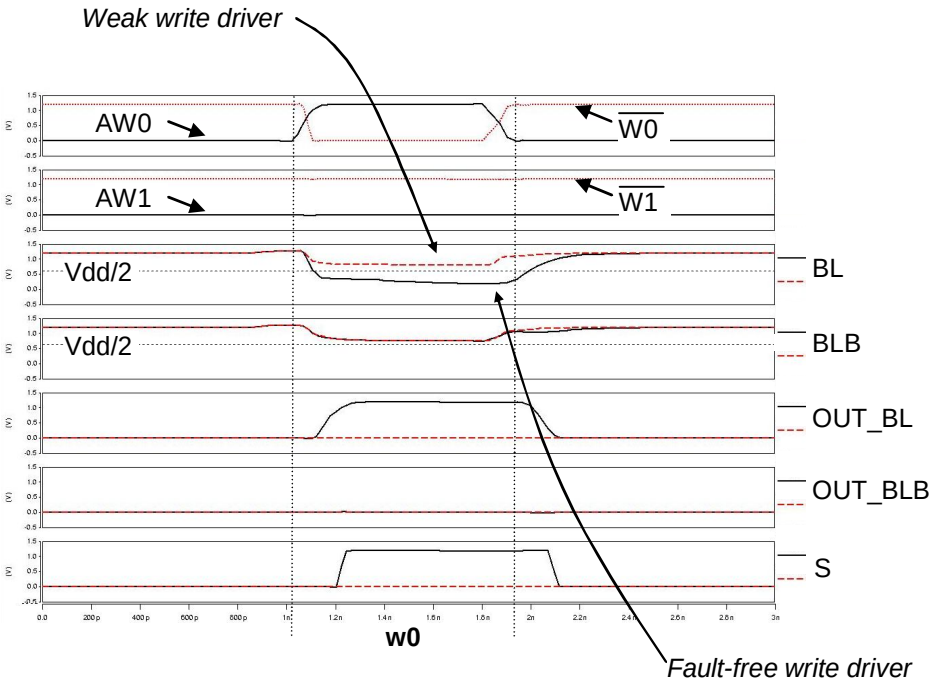

Figure II.4 – Diagnosis module functioning

Although efficient, such a structure is only able to verify if one bit line has a level lower than $Vdd/2$ and the other has a level higher than $Vdd/2$, irrespective of the type of write operations. Additional logic must therefore be added to distinguish between $w1$ and $w0$ operations as presented in the next sub-section.

II.1.3.2. Hardware diagnosis solution for the logic condition

Based on the previous comment, we must adapt the logic condition (see *Eq. II.1*) so that it can distinguish between $w0$ and $w1$ logic levels on bit lines. The solution we propose consists in comparing the bit line logic levels with the data to be written (node DataIn). The new logic condition becomes:

$$(\overline{BL} \oplus DataIn) \cdot (\overline{BLB} \oplus \overline{DataIn}) = 1 \qquad \textit{(Eq. II.3)}$$

It results on some modifications in the initial hardware implementation presented in Figure II.4, especially on the data processing part as shown in Figure II.5.

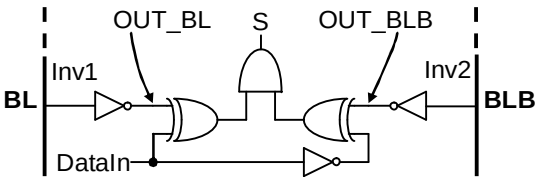

Figure II.5 – Data processing part of the diagnosis module

Waveforms in Figure II.6 show the simulation results of a faulty write driver which is not detectable with the initial data processing module. This faulty write driver always performs $w0$ operations even if it is configured to perform a $w1$ operation.

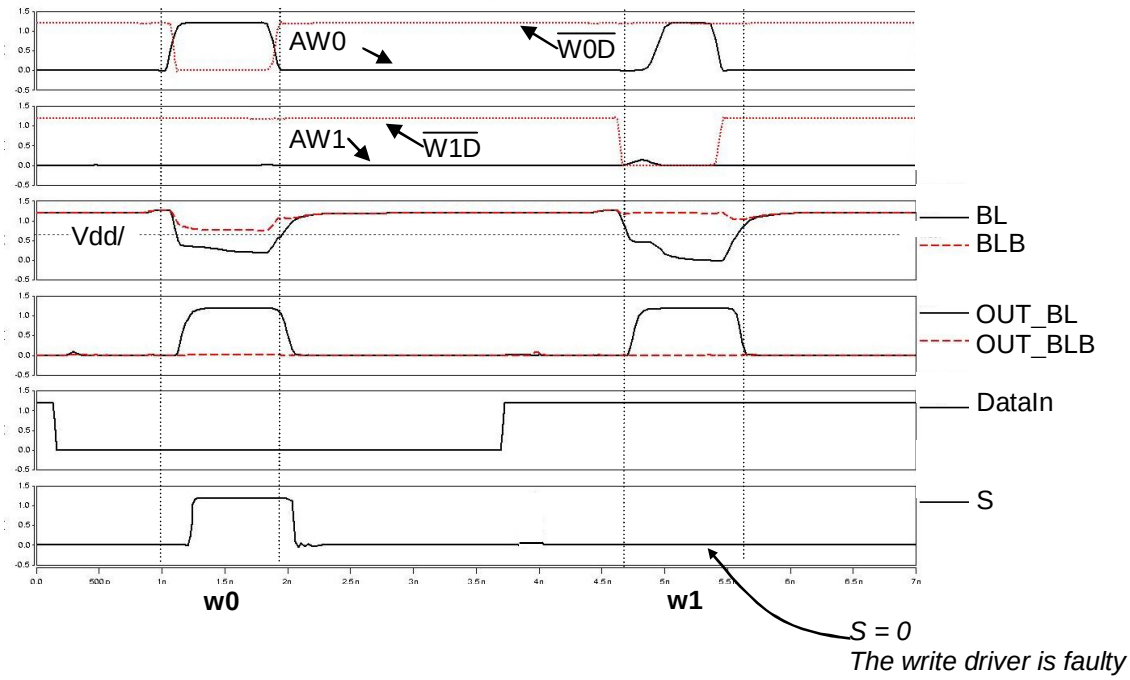

Figure II.6 – Simulation results of a faulty write driver

In Figure II.6, a $w0$ operation is first performed. BL becomes lower than $Vdd/2$ while BLB remains higher than $Vdd/2$. Node S is at logic '1' indicating that the write driver is fault-free. Then, the write driver is configured to perform a $w1$ operation, *i.e.* node DataIn is set to a logic '1'. As the driver can always perform $w0$ operations, node AW1 remains at logic '0' while AW0 = 1. As can be seen in Figure II.6, BL is lower than $Vdd/2$ and BLB remains higher than $Vdd/2$. In such case, the initial data processing part (see Figure II.3) would provide a logic '1' on node S indicating a fault-free write driver. With modifications presented in Figure II.5, node S provides a logic '0' that corresponds to a faulty write driver.

The DFD solution is effective and represents less than 0.5% of area overhead for a 512x512 SRAM.

II.1.3.3. Diagnosis sequence

The proposed diagnosis module is able to verify the logic (*Eq. II.3*) and analog (*Eq. II.2a* and *Eq. II.2b*) conditions. Obviously, a $w0$ and $w1$ operations are needed to

diagnose the write driver. In this case, only defects involving a static behavior will be diagnosed.

As mentioned in many published studies, defects in VDSM technology may also induce dynamic behaviors. Resulting fault models are dynamic faults [VDG00, ARS01, HAM03] as those that may affect the write driver (see previous part on test of dynamic faults in write drivers). In this study it is shown that two successive opposite write operations must be performed to detect dynamic fault that may affect the write driver. Consequently, the diagnosis sequence able to deal with static and dynamic faulty behaviors as well is the following:

$$wx\ w\bar{x}\ wx$$

So, only three operations are needed to fully identify a faulty or weak write driver. Reading the data on node S provides the required information on the correctness of the write drivers.

II.1.4. Description of the voltage-based DFD solution

In the previous section, we have defined equations *Eq. II.2.a* and *Eq II.2.b* and translate them into a design for diagnosis solution. Now, we consider *Eq. II.2.a bis* and *Eq. II.2.b bis*. Parameters β make the proposed solution tunable as the user can adapt them depending on the memory technology and the desired reliability level. As done for the previous DFD solution, we consider a 65nm SRAM technology and we have chosen parameters as follows:

- $\beta_1 = 0.1$
- $\beta_2 = 0.7$

Consequently, the write driver will be considered as faulty (“too much” weak) if it cannot discharged BL at a voltage lower than 10% of Vdd and maintain BLB at a voltage level higher than 70% of Vdd.

These logic and analog conditions can be translated into a DFD solution for SRAM write drivers as shown in the following section.

The proposed DFD solution accurately determines the analog levels on both bit lines during write operations. In the following sub-sections we first briefly describe the principle of the proposed solution and then, we provide the complete DFD structure and the corresponding diagnosis sequence.

II.1.4.1. DFD principle

As shown in Section II.1.2, the analog condition consists in verifying the final voltage level on both bit lines. A straightforward solution should consist in implementing a differential amplifier with a reference voltage connected to an input as presented in Figure II.7.

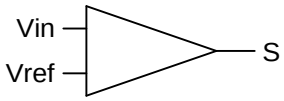

Figure II.7 – DFD principle

In the literature, different architectures are proposed to implement a differential amplifier and a voltage source. Each of them has its own specificities (area, accuracy, robustness, response time...). The selection depends on the application requirements.

The next sub-sections are dedicated to the description and implementation of the differential amplifier and the voltage source.

II.1.4.2. Implementation of the differential amplifier

The differential amplifier has to translate a weak differential voltage into a full swing differential voltage as soon as a diagnosis launch signal is activated. The resulting voltage level signal has to be saved until the end of the diagnosis phase. The amplification must therefore be instantaneous and not linear as performed with operational amplifiers for example. Consequently, such a requirement allows orienting our choice toward the sense amplifier already presented in the first part of the thesis (see Figure I.22 on part I).

As previously explain, such a sense amplifier is already used in SRAM to perform read operations. It has to translate a weak differential voltage between both bit lines into a full swing differential signal transmitted as a logic output data. So, a first question should be: in order to save area, why not reusing the existing sense amplifier for the diagnosis purpose?

Reusing this part of the memory requires many modifications on the I/O structure as the sense amplifier and the write driver are strongly correlated in a SRAM as can be seen in Figure II.8. In order to make the existing sense amplifier able to perform the diagnosis task, *i.e.* to sense the voltage levels on bit lines, we must add circuitry as shown in gray in Figure II.8. First, two voltage sources (V_{REFL} and V_{REFH}) are added on one input of the sense amplifier. Note that these voltage sources must be isolated from the sense amplifier during the

normal functioning mode by using pass gates. Selection of one of them depends on which level (high or low) we have to diagnose. In addition, each output of the write driver (WD and WDB) must be compared to the selected reference voltage source. Direct paths must be implemented depending on which write operation ($w0$ or $w1$) has to be diagnosed. Consequently, with such a principle, we can only diagnose one level at a time as there is only one sense amplifier able to compare one reference voltage source with one write driver output. Beside this additional circuitry, memory control signals must be modified in order to allow disabling the multiplexer controlled by the SEL signal and then isolate the sense amplifier from the memory.

All these modifications are difficult to implement as:

- they impact memory control signals,
- they impact write paths and
- they do not allow to diagnose low and high levels at the same time as only one sense amplifier is used.

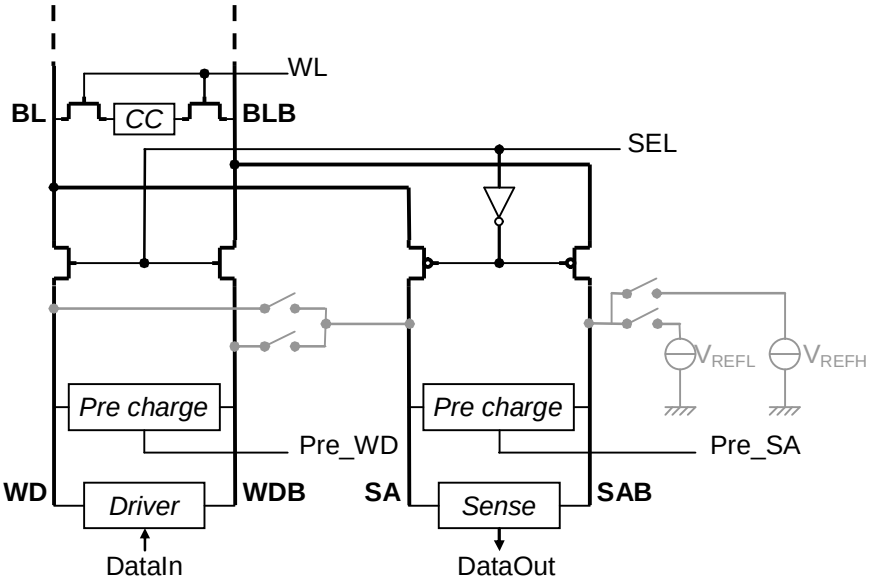

Figure II.8 – SRAM I/O circuitry

Consequently, a better solution consists in designing a DFD module that does not modify any control signal or write path and enable the diagnosis of both low and high levels at the same time. It can easily be realized with two additional sense amplifiers dedicated to a diagnosis purpose and connected to each write driver outputs (WD and WDB). Figure II.9 shows the resulting implementation. The connections between the write driver outputs and the sense amplifiers (SAL and SAH) are performed by N-type transmission gates (*MtnpgWD* and

MtnpgWDB) for diagnosing low level weak signals and by P-type transmission gates (*MtppgWD* and *MtppgWDB*) for diagnosing high level weak signals. These path gates are controlled by W0S and W1S signals which allow determining if we have to diagnose a $w0$ or a $w1$ operation.

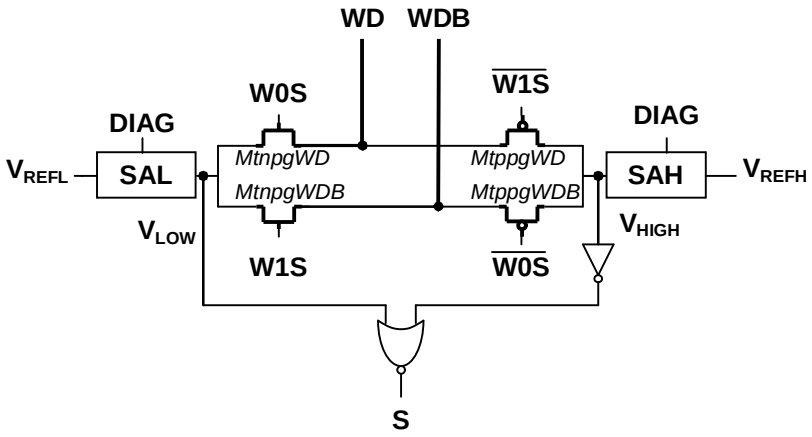

Figure II.9 – Hardware implementation of the DFD solution

Each sense amplifier receives the voltage level of the write driver (WD or WDB) on one input and a reference voltage (V_{REFL} = 10% Vdd and V_{REFH} = 70% Vdd) on the other. The DIAG signal allows activating the sense amplifiers during the diagnosis phase. Levels V_{LOW} and V_{HIGH} are the resulting amplification provided by each sense amplifier. These signals verify the analog condition.

The final diagnosis result (S) is a function (NOR gates) of both outputs V_{LOW} and V_{HIGH} levels in order to verify the logic condition. Table II.1 provides the truth table of the structure. A fault-free behavior is observed (S = 1) if WD is below 10% of Vdd, *i.e.* final V_{LOW} level is low, and WDB is above 70% of Vdd, *i.e.* final V_{HIGH} level is high (gray line on Table II.1) in case of a $w0$ operation. Consequently, such a DFD solution allows the diagnosis of low and high levels at the same time as two sense amplifiers and two reference voltage levels are embedded in the structure.

V_{LOW}	V_{HIGH}	S
0	0	0
1	0	0
0	**1**	**1**
1	1	0

Table II.1 – Truth table of the DFD module

Remark: *For the sake of clarity, we have considered that only one bit line is connected to one I/O circuitry (and hence only one write driver). Actually, more than one bit line (at least four) is connected to the same I/O circuitry. This means that each DFD module will be shared by several bit lines, thus decreasing the final area overhead (about 0.5% of area overhead for a 512x512 SRAM).*

Waveforms in Figure II.10.a and Figure II.10.b show the simulation results of the proposed DFD module for a fault-free and a weak write driver respectively. On both simulations, a $w0$ operation is performed. WD node is pulling down correctly in case of a fault-free write driver (Figure II.10.a). In Figure II.10.b, node WD does not reach Gnd as the write driver is weak. Consequently, when the DFD module is activated (DIAG = 1) two scenarios are observed:

In case of a fault-free write driver (Figure II.10.a), $V_{LOW} < V_{REFL}$ and $V_{HIGH} > V_{REFH}$ thus implying V_{LOW} = 0 and V_{HIGH} = 1. Output S of the DFD module provides a logic '1' meaning that the write driver is fault-free.

In case of a weak write driver (Figure II.10.b), $V_{LOW} > V_{REFL}$ and $V_{HIGH} > V_{REFH}$ thus implying V_{LOW} = 1 and V_{HIGH} = 1. Output S of the DFD module provides a logic '0' meaning that the write driver is faulty.

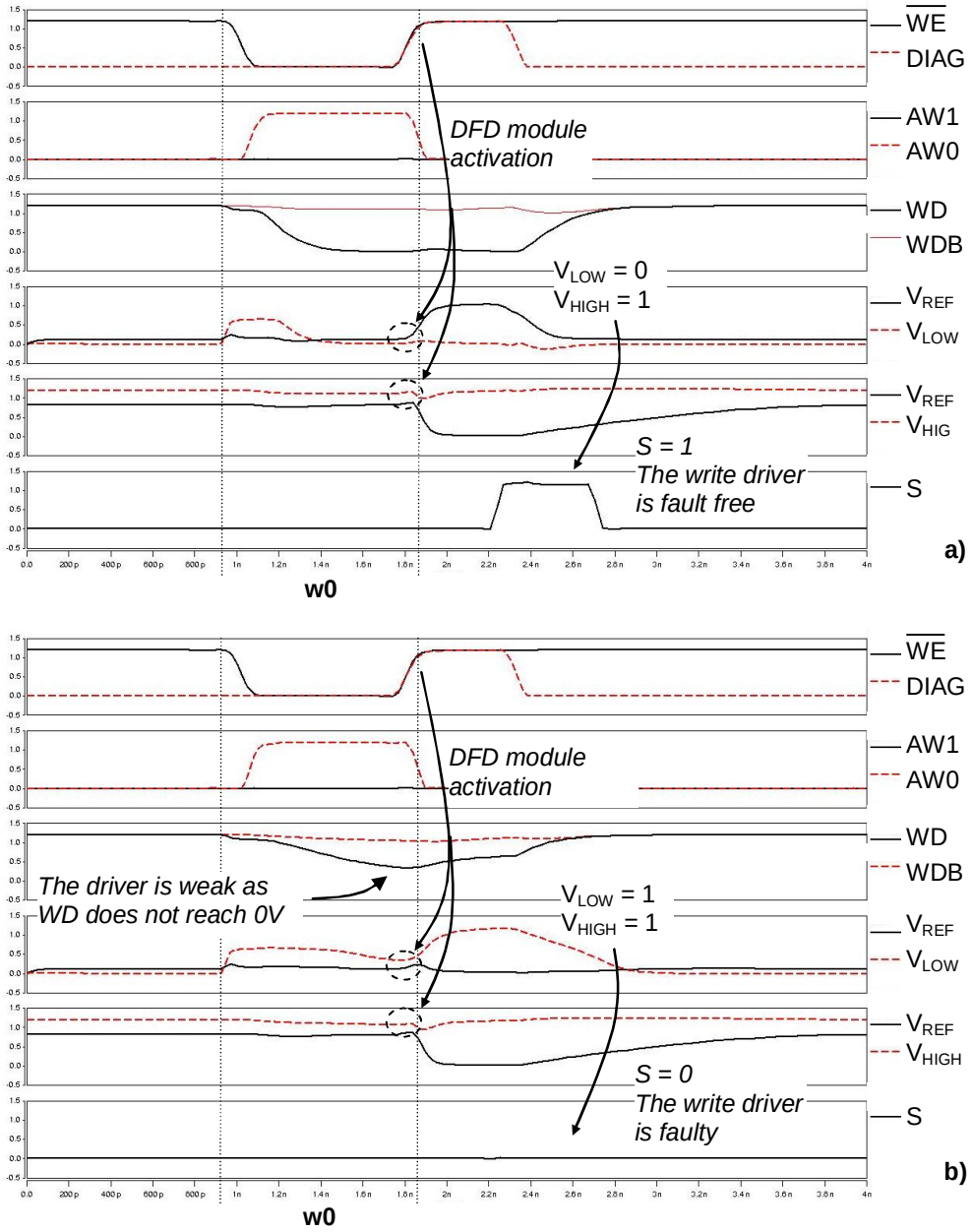

Figure II.10 – DFD module functioning for a) a fault-free and b) a weak write driver

II.1.4.3. Diagnosis sequence

The proposed diagnosis module is able to verify the logic (*Eq. II.3*) and analog (*Eq. II.2.a bis* and *Eq. II.2.b bis*) conditions. Obviously, a *w*0 and a *w*1 operations are needed to diagnose the write driver. In this case, only defects involving a static behavior will be diagnosed.

As shown for the current-based solution, the diagnosis sequence able to deal with static and dynamic faulty behaviors as well is the following:

$$wx\ w\bar{x}\ wx$$

So, only three operations are needed to fully diagnose a wrong or weak write driver.

II.1.5. Conclusions

In this chapter we have proposed two low cost DFD solutions for SRAM write drivers. They allow to identify wrong or weak write drivers by verifying logic and analog conditions that guarantee the write driver fault-free behavior. Moreover, they allow a fast diagnosis (only three write operations are needed) and induce a low area overhead (about 0.5% for a 512x512 SRAM).

The first solution, based on a current compensation, is simply dedicated to diagnosis as it allows to track weak write drivers in a pass/fail way according to a single threshold. In other words, it is not possible to perform any characterization purpose with such solution as the threshold detection is fixed by the design of the DFD module. In addition to diagnosis abilities, the second solution, using voltage sources and sense amplifiers, can be useful for write drivers characterization. Actually, voltage sources can be externally controlled by an Automatic Test Equipment (ATE) allowing the ability to monitor the voltage source and apply many different threshold voltages. Consequently, according to the targeted application, the user has to choose the most appropriate diagnosis solution.

Chapter 2. Software-based diagnosis solution

The usual software-based techniques for memory diagnosis are mainly based on signature analysis. They consist in creating a fault dictionary that is used to determine the correspondence between the signature and the fault models affecting the memory. The effectiveness of such diagnosis methods is therefore strictly related to the fault dictionary accuracy. To our knowledge, most of existing signature-based diagnosis approaches targets static faults only. In this chapter, we present a new diagnosis approach that represents an alternative to signature-based approaches. This new diagnosis technique, named history-based diagnosis, makes use of the effect-cause paradigm already developed for logic design diagnosis. It consists in creating a database containing the history of operations (read and write) performed on a faulty memory core-cell. This information is crucial to track the root cause of the observed faulty behavior and it can be used to generate the set of possible FPs representing the set of suspected faults. This new diagnosis method is able to identify static as well as dynamic faults. Although applied to SRAMs, it can be effective also for other memory types such as DRAMs.

This chapter is organized as follows: A state-of-the-art is first provided where we present basic signature-based diagnosis solutions, their functioning and drawbacks. The second Section presents an extension of these techniques able to consider not only static faults, but dynamic faults as well. In the third Section, our new history-based diagnosis is described. Principle, examples and results are provided. Finally, perspectives and conclusion are given in the last section.

II.2.1. State-of-the-art: signature-based diagnosis

Existing diagnostic methods are generally based on the *cause-effect* principle. In this section we propose to explain that, and then show its main drawbacks. A typical *cause-effect* diagnosis method is depicted in Figure II.11.

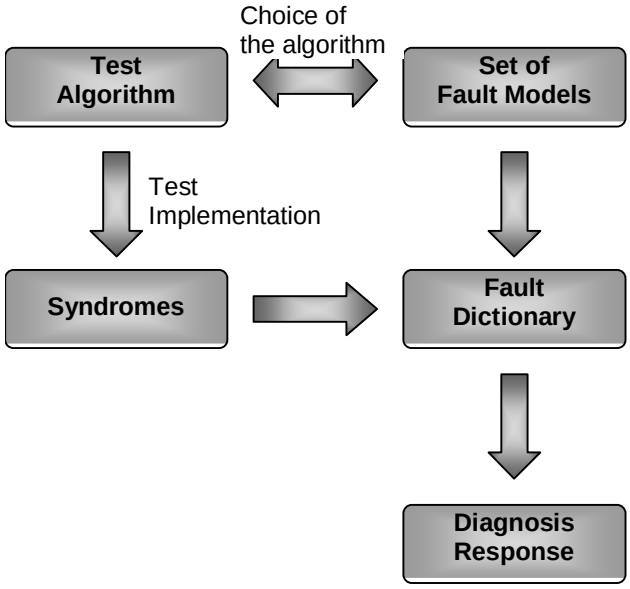

Figure II.11 – A *cause-effect* diagnosis method

Classical diagnostic techniques for memories are based on signatures analysis. The signature, also called syndrome, is composed of a set of read operations included in the considered March test. Each signature, representing a set of possible fault models affecting the memory, is collected in a dictionary.

As mentioned above, the quality of a diagnosis is given by the Diagnosability Ratio (DR), defined as the ratio between the number of distinguishable fault types and the number of total detectable fault types. Since the fault dictionary is based on a given March test, the DR is strictly related to:

- The set of fault models covered by the implemented March test.
- The number of read operations operated by the implemented March test.

To illustrate the signature-based diagnosis principle, let us consider the well-known March C-, whose structure is shown again in Figure II.12.

$$\updownarrow (w0) \uparrow (r0, w1) \uparrow (r1, w0) \downarrow (r0, w1) \downarrow (r1, w0) \updownarrow (r0)$$

Figure II.12 – March C- algorithm

For a given test algorithm, the fault dictionary can be generated by listing the fault models and their corresponding syndromes. For example, the fault dictionary, limited to stuck-at (SAF) and transition (TF) faults, for March C-, is given in Table II.2 [LI01].

Fault model	R_0	R_1	R_2	R_3	R_4
SAF0	**0**	**1**	**0**	**1**	**0**
SAF1	**1**	**0**	**1**	**0**	**1**
TF1	**0**	**1**	**0**	**1**	**0**
TF0	**0**	**0**	**1**	**0**	**1**

Table II.2 – Partial fault dictionary related to March C-

In Table II.2, R_i = 0 (1) means that the i^{th} read operation of the test algorithm has returned a correct (faulty) value for a specific memory core-cell. For example, a SAF0 corresponds to the failure of all $r1$ operations. Consequently, the March syndrome for SAF0 is (01010), as presented in Table II.2. It is important to mention that the signature does not depend on the faulty memory core-cell (*i.e.*, each faulty cell affected by a SAF0 has the same signature). Note that this fault dictionary can be extended to the whole set of fault models detected by March C-. Consequently, faulty test responses collected during March test application are used as pointer in the fault dictionary to provide the list of suspected faults.

Based on this principle, most of existing studies on memory fault diagnosis target static faults such as SAF, TF and CF [APP06, VAR06, HAR07]. These studies propose the extension of the considered March test by the addition of extra read operations, in order to increase the signature fields and therefore improve the DR. The first drawback of such techniques is that the increased complexity of March tests, *e.g.* the March DSS depicted in Figure II.13 of a 46N complexity in [HAR07], can be excessive to be used for industrial purpose. In addition, these solutions are most of the time unable to distinguish between all faults (or all fault models) and hence do not allow to determine which memory component is defective.

On the other hand, dynamic faults have been considered for diagnosis purpose in the literature only in [THA6], where the authors focus on dynamic CFs and extend the syndrome using the written data as field. Consequently, there is a clear need of new diagnosis solutions that consider both static and dynamic faults.

$$\{\uparrow(w0) \uparrow(r0,r0,w0,r0,w1) \uparrow(r1,w1,r1,w0)$$
$$\downarrow(r0,w0,r0,w1) \downarrow(r1,w1,r1,w0,w0,r0)$$
$$\downarrow(r0,w0,r0) \downarrow(r0,w1,r1) \downarrow(r1,w1)$$
$$\uparrow(r1,w0,r0) \uparrow(r0,w1,r1) \uparrow(r1,w0)$$
$$\downarrow(r0,w0) \uparrow(r0,w1) \downarrow(r1,w1,r1)$$
$$\downarrow(r1,w0) \uparrow(r0)\}$$

Figure II.13 – March DSS

II.2.2. Signature extension for dynamic fault diagnosis

In this Section, we introduce an extension of the signature technique, by adding new fields in the syndrome, to make it able to deal with dynamic faults as well. This extension is made possible by using information on the addressing sequence during the March test execution. The addressing order information has been demonstrated to be important in the detection of dynamic faults in SRAMs as well as the data background [DIL04a] that we intend to consider here. The proposed approach allows to diagnose dynamic faults, to distinguish between static and dynamic faults, and to localize the related failure in the memory. The additional information introduced in the signature is taken from the algorithm itself, thus it does not increase its complexity. Here, we illustrate the proposed signature-based diagnosis approach by considering as case study the dynamic fault Un-Restored Write fault (URWF), affecting write driver and pre-charge circuit of SRAMs.

II.2.2.1. Signature-based dynamic fault diagnosis

The proposed approach is still based on the classic signature methodology, but it reaches a high DR without raising the March test (MT) complexity, *i.e.* without adding additional read operations in the MT. For this purpose, we expand the number of the signature fields by adding information related to the address sequence used during the algorithm execution.

Considering all the fields in the signature, the information required for the diagnosis is the following:

- The tester report: faulty cell addresses.
- The executed MT.
- The list of fault models covered by the executed MT.
- The addressing sequence adopted during the MT execution.
- The SRAM architecture providing information about the core-cell array structure from the logic point of view.

In order to achieve a more efficient diagnosis, we now improve the signature-based diagnosis by introducing information extracted from the addressing order adopted during the MT execution and the SRAM architecture. For an easy understanding of our proposition, we consider the dynamic fault called Un-Restored Write Fault (URWF) as case study. The URWF can be due to different electric causes such as resistive-open defects in the write driver (see Part I) or in the pre-charge circuit [DIL05b] of SRAMs. The common effect in both cases is that the final voltage level of bit lines is erroneous at the end of the pre-charge phase, and the following read operation fails.

Figure II.14 depicts a simplified scheme of an SRAM composed by two 4x4 blocks. Each block presents its own I/O circuitry that is shared by four columns. As presented in Part I, the URWF resulting from a resistive-open defect in the write driver requires the following sequence of operations to be detected:

$$wx_{CCA}\, r\bar{x}_{CCB}$$

Both operations have to be performed on two distinct core-cells that belong to the same I/O circuitry (see Figure II.14).

When an URWF is due to a resistive-open defect in the pre-charge circuit, the sensitization sequence is the same than that described above, but in this case, both operations have to be performed on two distinct core-cells belonging to the same column.

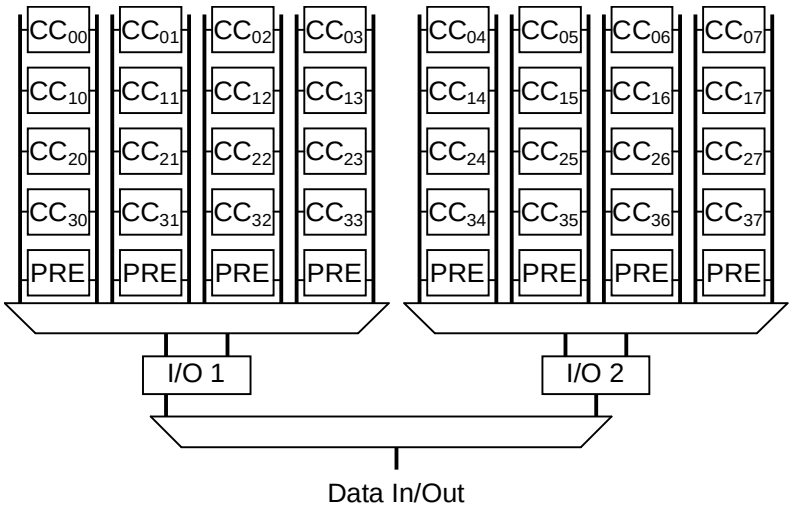

Figure II.14 – A two blocks SRAM architecture

Among March test algorithms, we have seen that March C- is able to detect both types of URWF if it is executed with the specific addressing order 'column after column'.

Considering the memory architecture shown in Figure II.14, we can determine the whole set of possible addressing situations:

- Ad_i is the address of the currently accessed cell and during a read at this address, a fault is detected.
- Ad_{i-1} is the address of the cell previously accessed with respect to Ad_i
- Ad_{i+1} is the address of the next cell to be accessed with respect to Ad_i .

Considering three consecutive address locations (Ad_{i-1}, Ad_i and Ad_{i+1}) during test execution, the possible combinations are the following ones:

- Ad_{i-1} belongs (or not) to the same I/O circuitry with respect to Ad_i.
- Ad_{i-1} belongs (or not) to the same column with respect to Ad_i.
- Ad_{i+1} belongs (or not) to the same I/O circuitry with respect to Ad_i.
- Ad_{i+1} belongs (or not) to the same column with respect to Ad_i.

Combinations described above are summarized in Figure II.15. On the left side, the list of the different addressing configurations concerning the previous accessed cell Ad_{i-1} is presented. The next accessed core-cell Ad_{i+1} is considered on the right part of the scheme.

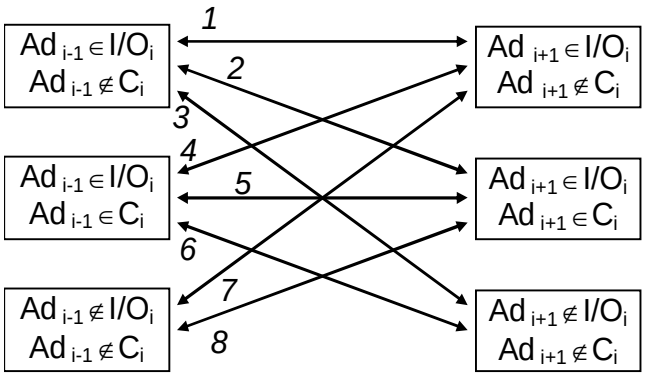

Figure II.15 – Possible address sequence during test execution for URWF detection, considering the memory architecture

For URWF detection, the sensitization sequence has to be applied at least on two distinct core-cells belonging to the same I/O circuitry (in case of write driver failure) or to the same column (in case of a faulty pre-charge circuit). Moreover, we consider not only the previous accessed core-cell but also the next accessed core-cell because most of March algorithms have up (↑) and down (↓) addressing order. Consequently, the next accessed core-cell in the up (↑) addressing order is the previous accessed core-cell during the down (↓) addressing order, and vice versa.

An URWF can be sensitized when two core-cells are accessed with any addressing order described in Figure II.15. The knowledge of the addressing sequence allows to deduce the following important information:

1. It allows to determine the faulty memory element, *i.e.* pre-charge circuit or write driver.
2. It allows to determine the nature of the observed fault, *i.e.* static or dynamic.

Let us consider the first point ‘1.’. In accordance with the possible addressing configurations presented in Figure II.15, four cases are possible:

- The configuration allows detecting URWFs caused by malfunction in a write driver.
- The configuration allows detecting URWFs caused by malfunction in a pre-charge circuit.
- The configuration allows detecting URWFs caused by malfunction of both pre-charge circuit and write driver but it cannot provide the failure localization.

- The configuration allows detecting URWFs caused by malfunction in both pre-charge circuit and write driver but also provides the failure localization.

In Table II.3, we list all possible *extended signatures* obtained with the application of the March C- algorithm (*c.f.* in Figure II.14) and leading to URWFs detection.

Fault model	Faulty element	CONF	R_0	R_1	R_2	R_3	R_4	Ad_{i-1}	Ad_{i+1}
URWF	WD	1	1	0	1	0	0	10	10
	WD	2a	1	0	1	0	0	10	11
	Pre	2b	0	0	1	0	0	10	11
	WD	3	1	0	0	0	0	10	00
	WD	4a	1	0	1	0	0	11	10
	Pre	4b	1	0	0	0	0	11	10
	WD, Pre	5	1	0	1	0	0	11	11
	WD, Pre	6	1	0	0	0	0	11	00
	WD	7	0	0	1	0	0	00	10
	WD, Pre	8	0	0	1	0	0	00	11

Table II.3 – List of *extended signatures* for URWF detection during March C- execution

The two first columns provide the fault model (URWF) and the memory element(s) whose failure involves the URWF: *WD* for write driver and *Pre* for pre-charge circuit. The third column gives the configuration, with respect to the scheme in Figure II.15 (labels on arrows). Columns from four to eight provide the classical March C- signatures for URWF detection. Finally, the two last columns of Table II.3 are fields we have added to represent the addressing order, Ad_{i-1} for the previous accessed core-cell and Ad_{i+1} for the next one. These additional fields require two bits to represent all address configurations presented in Figure II.15:

- The first bit indicates if address Ad_{i-1} (or Ad_{i+1}) shares the same I/O element than the current accessed memory core-cell (‘1’ if yes, ‘0’ if no, x if don’t care).
- The second bit indicates if address Ad_{i-1} (or Ad_{i+1}) shares the same column than the current accessed memory core-cell (‘1’ if yes, ‘0’ if no, x if don’t care).

For example, Ad_{i-1} = 10 means that the previous accessed core-cell shares only the same I/O circuitry with the current accessed core-cell. For the diagnosis of other dynamic fault models, it is necessary to use additional bits to describe other specific addressing sequence. For example, Ad_{i-1} and Ad_i belonging to the same word line is the addressing sequence useful to detect dRDF, unitary hamming distance addresses are necessary to detect Address Decoder Open Faults (ADOF) [DIL04a].

For a better understanding of Table II.3, we propose the reading of the first line. The last two columns indicate that Ad_{i-1} and Ad_{i+1} belong to the same I/O of the current accessed core-cell i, but not to the same column (Ad_{i-1} = Ad_{i+1} = 10). With such addressing sequence, the only detectable failing element is the write driver (WD), as previously explained; this information is shown in column two. Finally, the March C- application with this addressing configuration between Ad_{i-1}, Ad_i and Ad_{i+1} provide the basic signature based on read operations (10100), exposed in columns three to eight. Thus, the extended signature is '101001010'.

The list of extended signatures presented in Table II.3 shows that:

- The addressing configurations 1, 3 and 7 allow the detection and the localization of URWF in the write driver.
- The addressing configurations 5, 6 and 8 allow the detection of URWF but do not provide any information on the failure localization. That means the URWF can be due to a failure in the pre-charge circuit or in the write driver as well.
- The addressing configurations 2 and 4 allow the detection of URWF but also are able to exactly determine the failure localization. In fact, with the same addressing sequence, different syndromes are generated according to the faulty elements. The 'a' suffix is attached to faulty write drivers, whereas the 'b' suffix is attached to faulty pre-charge circuits.

In the signature, the additional fields concerning the addressing sequence have been helpful to determine the occurrence of an URWF as well as the failure localization.

Now, we analyze the second point mentioned above '2.', *i.e.* how to determine the static or dynamic nature of the fault. In Table II.4, we give the set of signatures related to URWF and CFst (taken from [LI01]), considering March C-execution.

Fault model	R0	R1	R2	R3	R4	Ad $_{i-1}$	Ad $_{i+1}$
URWF	1	0	0	0	0	10	00
	1	0	0	0	0	11	10
	1	0	0	0	0	11	00
CFst(L,1,1)	1	0	0	0	0	xx	xx

Table II.4 –Signatures -URWF vs. CFst -

In this table, the fields marked with 'x' associated to the addressing configuration CFst signature, mean that the addressing order has no impact on the fault detection. Table II.4 shows that URWF and CFst are not distinguishable as they have the same syndrome. However, it also shows that depending on the sequence of accessed core-cells during test application, we can state if there is no URWF occurrence. In other words, with a single test sequence application, it is possible to determine if the memory is affected by static faults only. However, further test applications with different addressing sequences should be useful to completely differentiate static and dynamic faults.

II.2.2.2. Discussions

In this Section, we have proposed an approach for dynamic fault diagnosis in SRAMs. This approach is based on the extension of existing signature-based diagnosis methods. We show that tacking in account information concerning the addressing configuration of the executed March test can be crucial for the diagnosis of dynamic faults. We have demonstrated the effectiveness of the proposed solution in identifying the failure location in the memory on a case study, the URWF.

However, such technique presents many drawbacks. First, these solutions based on the extension of the signature suffer from limitations due to the use of an established fault dictionary. Secondly, as all possible memory configurations have to be take into account for the complete description of a fault model, the store information in the dictionary may become too high when considering the whole set of static and dynamic faults.

Consequently, there is a clear need of new diagnosis solutions that:

- consider the whole set of static and dynamic faults
- are able to provide accurate location of the faulty component (in the core-cell array, write drivers, sense amplifiers, address decoders, pre-charge circuits, etc.)
- are not limited to the *a priori* knowledge of the targeted faults, but that generate dynamically the diagnosis response.

II.2.3. History-based diagnosis

In this section, we present a new diagnosis approach that represents an alternative to signature-based approaches. This new diagnosis technique is based on the *effect-cause* paradigm already developed for logic design diagnosis [ABR84]. It consists in creating a database containing the history of operations (read and write) performed on those core-cells, where read operations have returned faulty logic values, during the test phase. This information is crucial to track the root cause of the observed faulty behavior and is used to generate the set of possible FPs [VDG00] representing the suspected fault models.

Such history-based diagnosis approach offers many advantages. It does not require the *a priori* knowledge of the set of fault models targeted by the test algorithm because it does not rely on an established fault dictionary. It does not suffer from an additional limitation of signature-based approaches with respect to the treatment and storage of large data volume. Moreover, this method is able to perform the diagnostic of both static and dynamic faults and provides a better DR compared to signature-based diagnosis approaches. Another feature of the proposed history-based approach is its capability to provide accurate and reliable information on the fault location. This is imposed by the fact that some fault models can be related to multiple possible electric causes, leading to a difficult location of the faulty memory component (address decoders, core-cells, sense amplifiers, write drivers…). For example, it has been previously shown that an URWF can be due to defects locating in a pre-charge circuit or in a write driver of SRAMs. A signature-based diagnosis approach would indicate that the memory is affected by an URWF, without any information on the faulty component of the memory where the malfunction is actually caused. Conversely, with our history-based diagnosis approach, the diagnosis report will indicate that the memory is possibly affected by an URWF also specifying the suspected memory component (pre-charge circuit or write

driver). Such information is very helpful for the yield ramp up as well as to guide the repair schemes.

In order to perform a relevant number of experiments, we have created a dedicated software diagnosis tool. The achieved experimental results are analyzed and compared to results obtained with a classical signature-based approach. The efficiency of the history-based diagnosis in producing a list of suspected faults as well as the indication of the fault location is proven.

II.2.3.1. Principle

The principle of the history-based diagnosis is based on the collection of two types of relevant information:

- the faulty responses provided by the tester.
- the record of the sequence of preceding operations performed on the core-cells where read operations have returned faulty logic values during the test. With this information, a set of FPs is generated.

As can be seen in Figure II.16, the proposed diagnosis solution requires three inputs:

Memory Architecture: this input provides information related to the tested memory in terms of dimension (number of row and columns), I/O organization and other information about the structure.

March Test Specifications: this input provides information on the applied test algorithm in terms of sequence of operations performed on the memory and addressing order (row after row, named '*fast R*', column after column, named '*fast C*',…).

Tester Report: this input provides information about the results of the test. For each observed error, this report indicates which is (are) the read operation(s) that reveal the fault and the corresponding core-cell address.

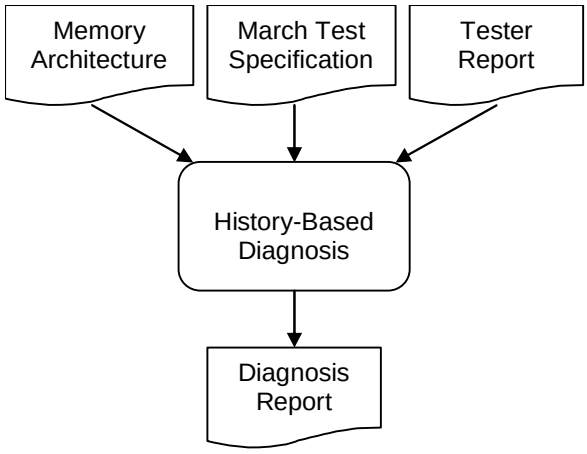

Figure II.16 – History-based diagnosis principle

As indicated in previous sections, our diagnosis solution does not require any list of fault models in input as in standard signature-based diagnosis solutions.

Let us now introduce the four steps of the history-based diagnosis procedure:

Step 1: History of faulty reads

Step 2: History of fault-free reads

Step 3: FP compilation

Step 4: Fault model allocation

These steps are explained in detail in the following sub-sections.

II.2.3.2. Step 1: History of faulty read operations

This first step of the proposed diagnosis process consists in recording the history of operations performed on the faulty core-cell. The history concerns only the back operations that lead to faulty read operations. Starting from a faulty read operation, we record all the operations previously performed on the affected core-cell until the last read operation.

Let us illustrate these principles with a hypothetical 4x4 SRAM, whose scheme is presented in Figure II.17.

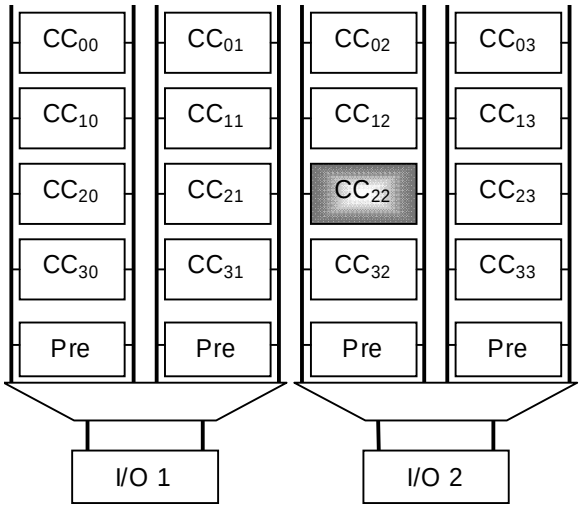

Figure II.17 – A 4x4 memory core-cell array

Let us assume that this memory is affected by a TF0 (Transition Fault 1 to 0) in core-cell CC_{22}. Consider that the applied March algorithm is the March C- with a *fast C* addressing order. The test application provides the following syndrome (available from the *tester report*):

$$0\,0\,1\,0\,1\;(Address\;CC_{22})$$

meaning that the $r0$ operations of March elements M3 and M5 performed on CC_{22} have returned a faulty value.

With available information (SRAM structure, MT and syndrome) we can start to generate the history record for each faulty read operation. The first faulty read operation is the $r0$ of the March element M3. The previous operations performed on CC_{22} were a correct $r1$ and a $w0$ of M2. The history record generation stops at the fist previous read operation. The resulting history record of the first faulty read is denoted as H_0 and is composed as follows:

$$H_0 \;=\; 1\,w0\;\;r0$$

where the logic '1' indicates the last value returned by the read in the core-cell (by the March element M2), followed by the $w0$ of M2 and the faulty $r0$ of March element M3. From H_0 we compute all possible FPs that can explain the faulty behavior on CC_{22}. We include these FPs in a set denoted as eFP_0 and composed as follows:

$$eFP_0 \;=\; \{(1\,w0),(0\,r0)\}$$

Note that in the previous expression, and in all the following ones of the same type, the notation of FP is simplified to the sensitization sequence ('S' in the formal notation [VDG00]). The FP '$1\,w0$' can be interpreted as a failing $w0$ on CC_{22}. The second FP, '$0\,r0$', contains the actual faulty $r0$.

In the same way, H_1, related to the second faulty read on CC_{22}, is computed and results in a new history record:

$$H_1 = 1\,w0\;\; r0$$

From H_1 we obtain eFP_1:

$$eFP_1 = \{(1\,w0), (0\,r0)\}$$

Assuming that a memory component (core-cell, write driver, pre-charge circuit,…) may be affected by a single fault, the root cause of the observed error has to be present in all sets of FPs. Consequently, the resulting eFP_{faulty} related to CC_{22} is simply obtained by intersecting eFP_0 and eFP_1:

$$eFP_{faulty} = \{(1\,w0), (0\,r0)\}$$

II.2.3.3. Step 2: History of correct read operations

In order to reduce the set of FPs in eFP_{faulty}, we have to exclude those FPs that certainly do not lead to a fault. For this purpose, we consider the FPs that are generated by non-faulty read operations. Considering again the example developed in Section II.2.3.2 (TF0 on core-cell CC_{22})

According to the obtained signature (00101), one $r0$ operation among three is correct. Consequently, we build the history of this $r0$ as follows (in case of more than one correct $r0$, a history record should have been generated for each correct $r0$):

$$H_0' = X\;\; w0\;\; r0$$

with ‘X’ meaning that the contents of the core-cell is unknown before starting the March element M0.

From H_0’ we generate $eFP_{fault\text{-}free}$ as follow:

$$eFP_{fault-free} = \{(X\,w0), (0\,r0)\}$$

The FPs in $eFP_{fault\text{-}free}$ are those for which no faulty behaviors has been observed.

Note that the history of the two correct $r1$ is not considered because it could not reduce the number of FPs.

II.2.3.4. Step 3: FP Compilation

After Step 1 and Step 2, we have two sets of FPs: eFP_{faulty} and $eFP_{fault\text{-}free}$. Now, we have to remove from eFP_{faulty} the FPs composing $eFP_{fault\text{-}free}$. This is reported in the following equation:

$$eFP_{report} = eFP_{faulty} - (eFP_{faulty} \cap eFP_{fault-free}) \qquad \textit{(Eq. II.4)}$$

For the considered example with the occurrence of a Transition Fault TF0 on CC_{22}, we obtain:

$$eFP_{report} = \{(1\ w0)\}$$

This sets of FPs (in this case with only one element) represent the possible causes of the observed error.

II.2.3.5. Step 4: Fault Model Allocation

At this stage of the diagnosis process, we have the final report of FPs. From eFP_{report}, we associate the corresponding fault models to each FP as described in [VDG00]. Let us consider again the above examples, for which the reduced set of FPs is:

$$eFP_{report} = \{(1\ w0)\}$$

In this case, there is only one FP in the list and the corresponding fault model is a TF0 (Transition Fault 1 to 0), which corresponds to the actual fault affecting the memory.

II.2.4. Diagnosis of dynamic faults

The history-based diagnosis approach, as presented in the previous section, is able to diagnose static faults. Hereafter, we extend this new diagnosis approach to make it able to diagnosis dynamic faults. In sub-section II.2.4.1 we present a set of dynamic faults that we want to diagnose. For each of them we provide their definition and we highlight important conditions that will be helpful to apply our diagnosis technique. In sub-section II.2.4.2, we present how we improve our diagnosis approach in order to cover dynamic faults.

II.2.4.1. Dynamic fault models

At this stage of our study, we consider three two-cell dynamic faults and one single-cell dynamic fault. Additional dynamic fault models will be implemented in future developments.

Slow Write Driver Fault (SWDF): *A write driver is said to have a SWDF if it cannot act a w0 (w1) when this operation is preceded by a w1 (w0). This results in a core-cell that does not change its data content.*

This dynamic fault model is the consequence of resistive-open defects in the control part of the write driver. It involves an erroneous write operation when the write driver performs two successive write operations with opposite data values. A SWDF requires the following test sequence to be detected:

$$wx\ w\bar{x}\ r\bar{x} \qquad (1)$$

where the two write operations are for sensitization and the read operation for observation. Moreover, the two write operations have to be performed by the same write driver.

Considering that our diagnosis method consists in taking into account the operations performed on the faulty core-cell we must store additional information concerning the previous data written by the write driver. Consequently, we formulate the first requirement that will be exploited by our diagnosis method as follows:

Requirement 1: *During the diagnosis process we must take a record of the data previously written by the write driver.*

Un-Restored Write Fault (URWF): *The pull up of one of the two bit lines is not completely achieved after the state reached with a write operation. Consequently the following read operation of an opposite data in a cell belongs the same I/O circuitry is not correctly acted [ADA97].*

Un-Restored Destructive Write Fault (URDWF): *The same definition as URWF but in addition to the faulty read operation, the cell flips.*

These two fault models are the consequence of resistive-open defects in the pre-charge circuit and/or the write driver. Both affect the read operation when it is preceded by a write operation. As presented in the first Part of the thesis, the URWF and URDWF resulting from a resistive-open defect in the write driver requires the following sequence of operations to be detected:

$$wx_{CCA}\ r\bar{x}_{CCB} \qquad (2)$$

where wx_{CCA} means write the value x in cell CC_A and $r\bar{x}_{CCB}$ means read the opposite value in cell CC_B. Both operations have to be performed on two distinct core-cells that belong to the same I/O circuitry.

As mentioned in [DIL05b], when an URWF is due to a resistive-open defect in the pre-charge circuit, the sensitization sequence is the same than (2), but in this case, both operations have to be performed on two distinct core-cells belonging to the same column, *i.e.* sharing the same pre-charge circuit.

As for SWDF, we formulate the second requirement that will be exploited by our diagnosis method as follow:

Requirement 2: *During the diagnosis process we must take a record of where (same column, same I/O) the last write operation has been performed.*

dynamic Read Destructive Fault (dRDF): *A cell is said to have a dRDF if a write operation immediately followed by a read operation performed on the core-cell changes the logic state of this core-cell and returns and incorrect value on the output [VDG00, HAM02].*

This dynamic fault model is related to resistive-open defects affecting the core-cell and it has been improved in [BOR03b] where the authors have shown that multiple read after the write operation may also induce the faulty swap of the targeted core-cell. Consequently, a dRDF requires the following sensitizations sequence:

$$wx\ rx\ rx\ rx\ rx \ldots rx \qquad (3)$$

In [DIL04b] it has been shown that operations performed on a core-cell involve a stress on the other core-cells belonging to the same word line. This stress, called Read Equivalent Stress (RES), is equivalent to a read operation.

For our diagnosis tool, we must therefore consider the record of such stresses in order to be able to deal with dRDF diagnosis. As for the others dynamic fault models exposed above, we formulate the third requirement that will be exploited by our diagnosis method as follows:

Requirement 3: *During the diagnosis process we must take a record of the sequence of consecutive read operations or RESs (Read Equivalent Stresses) undergone by the core-cell presenting a faulty read.*

II.2.4.2. Application

Based on requirements exposed above, we have to consider the record of additional information during the diagnosis process in order to cover dynamic faults. For this purpose, in the history record, we include an Additional Information Vector (AIV), which stores information about the previously accessed core-cell in address Ad_{i-1} (see Requirements 1 and 2), information about the next accessed core-cell in address Ad_{i+1} (see Requirements 1 and 2)

and all the RES that the faulty core-cell has undergone (see Requirement 3). This vector contains the following information:

- One bit to report if Ad_{i-1} and Ad_i belongs to the same I/O (0: false, 1: true) → useful for URWF (1).
- One bit to report if Ad_{i-1} and Ad_i belongs to the same column (0: false, 1: true) → useful for URWF detection (2).
- The last operation ($r0$, $r1$, $w0$ or $w1$) performed on Ad_{i-1} just before the faulty read → useful for URWF (3).
- The number of break(s) between the faulty read operation on Ad_{i-1} and the last operation done on Ad_{i-1} → useful for URWF (4).
- A record of the previously data written by the write driver (0: fall transition, 1: up transition, -1: no transition) → useful for SWDF detection (5).

Table II.5 groups all these requirements and affects a number for each of them. The four first bits are related to the previous accessed core-cell, the next third concern the next accessed core-cell, and the last bit is associated with the write driver input.

Ad_{i-1}				WD
1	2	3	4	5
I/O	column	operation	break	Transition

Table II.5 – Additional Information Vector legend

Let us now illustrate the use of this AIV on three examples.

Example 1: dRDF

Let us assume that the core-cell CC_{22} (see Figure II.17) is affected by a dRDF, *i.e.* one $r1$ just after a $w1$ causing the faulty swap of CC_{22}. In this case, March C- is applied with *fast R* addressing order as proposed in [DIL04b]. The tester report presents the following syndrome:

$$0\ 1\ 0\ 1\ 0\ (Address\ CC_{22})$$

H_0 is the history of the first faulty read:

$$H_0 = 0\ w1\ RES^2\ B^{24}\ RES^4\ r1$$

The last read operation is a fault-free $r0$, so that the first term in H_0 is '0'. The next operation after this correct $r0$ is a $w1$ operation performed on CC_{22}, followed by a $r1$. All the

others terms in between represent what the core-cell electrically undergoes: B means a Break and RES means Read Equivalent Stress.

Let us first explain how we have obtained the first term (RES^2). During the application of March element M1, a $w1$ operation is performed on CC_{22}. Subsequently, the operations $(r0, w1)$ are acted on core-cell CC_{23}. As CC_{23} belongs to the same word line than CC_{22}, the latter undergoes two stresses denoted as RES^2.

Then, the $(r0, w1)$ operations are performed on the four core-cells of the last row of the core-cell array. Consequently, CC_{22} is not electrically stimulated during 8 clock cycles. Then, March element M2 is run. The $(r1, w0)$ operations are acted on all core-cells of the two first rows. Core-cell CC_{22} does not undergoes any electrical stimulation during 16 clock cycles. At the end, CC_{22} is not accessed, even indirectly (RES), during 24 clock cycles, thus we report B^{24}.

The $(r1, w0)$ operations are now performed on CC_{20} and CC_{21}. As these two core-cells belong to the same word line than CC_{22}, CC_{22} undergoes four stresses, reported with RES^4. At the same time, we compute AIV_0 as follows:

Ad_{i-1}				WD
I/O	column	operation	break	Transition
0	0	w0	0	-1

From H_0 and AIV_0 we obtain eFP_0:

$$eFP_0 = \{(0\,w1), (1\,r1), (0\,w1\,r1), (0\,w1\,r1\,r1), (0\,w1, 1\,r1)\}$$

The two first FPs of eFP_0 consider respectively the first and last operations of H_0 with '$0\,w1$' meaning that the $w1$ operation may have failed and '$1\,r1$' containing the actual faulty $r1$ operation.

Then FP_0 is completed with FPs related to the action of the RESs recorded in H_0. As the $w1$ operation on CC_{22} is immediately followed by two RES (RES^2), we obtain the '$0\,w1\,r1$' and '$0\,w1\,r1\,r1$' FPs (considering RES ≈ read operation). Finally, the last FP of eFP_0 is obtained with the help of the AIV_0. The '$0\,w1\,,1\,r1$' FP means that $w1$ is acted on a core-cell corresponding to Adi-1 and $r1$ is performed on the core-cell corresponding to Adi.

In the same way, we compute H_1 that concerns the second faulty read:

$$H_1 = 0\ w1\ RES^4\ B^{24}\ RES^2\ r1$$

and AIV_1 related to H_1 is the following:

Ad_{i-1}				WD
I/O	column	operation	break	Transition
1	0	**w0**	**0**	-1

From H_1 and AIV_1 we obtain eFP_1:

$$eFP_1 = \{(0\,w1), (1\,r1), (0\,w1\,r1), (0\,w1\,r1\,r1),$$
$$(0\,w1\,r1\,r1\,r1), (0\,w1\,r1\,r1\,r1\,r1), (0\,w1I/O\,, 1\,r1)\}$$

Note that the index 'I/O' in the last FP means that the $w1$ is performed on a core-cell belonging to the same I/O circuitry used by core cell that has recorded a faulty read.

eFP_{faulty} is obtained by intersecting eFP_0 and eFP_1:

$$eFP_{faulty} = \{(0\,w1), (1\,r1), (0\,w1\,r1), (0\,w1\,r1\,r1)\}$$

This time, all $r1$ operations have returned an incorrect data value, implying that $eFP_{report} = eFP_{faulty}$:

$$eFP_{report} = \{(0\,w1), (1\,r1), (0\,w1\,r1), (0\,w1\,r1\,r1)\}$$

The last step of the method assigns the fault models to the FPs found. From eFP_{report} we obtain as fault candidates:

- TF1: this fault model is related to the FP '$0\,w1$'.
- SAF0, RDF, IRF: these fault models are related to FP '$1\,r1$'.
- dRDF (1 $r1$, 2 $r1$): This fault model is related to FPs '$0\,w1\,r1$' and '$0\,w1\,r1\,r1$'.

The dRDF (1 $r1$, 2 $r1$) means that the core-cell has swapped after one or two consecutive $r1$ operations just after the $w1$. This last fault model proposed in the diagnosis report corresponds to the one we have injected.

Example 2: URWF

As second example, we consider the Un-Restored Write Fault (URWF). Such a fault model is caused by defects in the pre-charge circuit or in the write driver. In this example, we inject a defect in the pre-charge circuit of the last column of the memory presented in Figure II.17.

After applying the March C- with a *fast C* addressing order, the test report presents the following syndromes:

$$0\,0\,0\,1\,0\;(Address\;CC_{03})$$

$$0\,1\,0\,1\,0\;(Address\;CC_{13})$$

$$0\,1\,0\,1\,0\;(Address\;CC_{23})$$

$$0\,1\,0\,0\,0\;(Address\;CC_{33})$$

It is important to notice that all the four core-cells belonging to the faulty pre-charge circuit provide faulty responses. Consequently, the diagnosis procedure has to be applied four times. Let us first detail the case of the faulty reads in core-cell CC_{03}. H_0 is the history of the faulty read:

$$H_0 = 0\ w1\ B^6\ RES^2\ B^6\ RES^2\ B^6\ RES^2\ B^6\ r1$$

and AIV_0 related to H_0 is the following:

Ad_{i-1}				WD
I/O	column	Operation	break	Transition
1	**1**	**w0**	**0**	-1

From H_0 and AIV_0 we obtain eFP_0:

$$eFP_0 = \{(0\,w1), (1\,r1), (1\,w0I/O, c\,, 1\,r1)\}$$

As there is only one faulty read, $eFP_{faulty} = eFP_0$.

Now, we build the history of the remaining correct read as follow:

$$H_0' = 0\ w1\ B^6\ RES^2\ B^6\ RES^2\ B^6\ RES^2\ B^6\ r1$$

and AIV_0' related to H_0' is the following:

Ad_{i-1}				WD
I/O	Column	Operation	break	Transition
1	0	**w0**	**0**	-1

From H_0' and AIV_0' we obtain eFP_0':

$$eFP_0' = \{(0\,w1), (1\,r1), (1\,w0_{I/O}\,, 1\,r1)\}$$

As there is only one correct read:

$$eFP_{fault-free} = eFP_0'$$

From the intersection of eFP_{faulty} and $eFP_{fault\text{-}free}$, we obtain eFP_{report} as follow:

$$eFP_{report} = \{(1\,w0_c\,, 1\,r1)\}$$

The fault model related to eFP_{report} is an URWF. In addition, as the previous $w0$ operation has been performed on the same column, we can state that the URWF is due to a defect in the pre-charge circuit (incorrect bit line pull up during the restoring phase between two operations).

The same procedure is followed for the remaining syndromes. We obtain for CC_{13} and CC_{23}:

$$eFP_{report} = \{(0\,w1), (1\,r1), (1\,w0_{I/O,c}\,, 1\,r1)\}$$

meaning that both core-cells may be affected by a TF1, SAF0, IRF or RDF. From the last FP of eFP_{report} we also conclude that an URWF may be the root cause of the observed errors. But this time we cannot provide any information on the defect location (I/O or pre-charge circuitry).

For the last syndrome related to CC_{33} we obtain:

$$eFP_{report} = \{(1\,w0_{I/O,c}\,, 1\,r1)\}$$

meaning that an URWF is the root cause of the observed error. Like the previous case, we cannot state on the defect location. Otherwise, an important result is that we have found out a unique fault model for all the four syndromes. The URWF related to a defect in the pre-charge circuit is present in all fault lists. Of course, this fault model has a much higher probability to be the actual and singular root cause of the malfunctions respect to TF1, SAF0, RDF, IRF and URWF (due to a defective write driver).

Example 3: SWDF

A third example concerns Slow Write Driver Fault (SWDF). Such a fault model is caused by defects in the write driver. In this example, we inject a defect in the write driver belonging to I/O1 (*c.f.* Figure II.17). After applying the March C- with a *fast C* addressing order, the test report presents the following syndromes:

$$0\,1\,0\,0\,0\;(Address\;CC_{00})$$

$$0\,0\,0\,1\,0\;(Address\;CC_{31})$$

As previously seen for URWF, the diagnosis procedure has to be applied twice (for each faulty core-cell). Let us first detail the case of the faulty reads in core-cell CC_{00}.

H_0 is the history of the faulty read:

$$H_0 = 0\ w1\ B^6\ RES^2\ B^6\ RES^2\ B^6\ RES^2\ B^6\ r1$$

and AIV_0 related to H_0 is the following:

Ad_{i-1}				WD
I/O	column	operation	break	Transition
0	0	w1	0	**1**

From AIV_0, we highlight the fact that the write driver has undergone an up transition (see last column of AIV_0). It means that the previous write operation done by the write driver before the $w1$ operation (see H_0) performed on CC_{00} was a $w0$.

Consequently, from H_0 and AIV_0 we obtain eFP_0:

$$eFP_0 = \{(0\ w1), (1\ r1), (1\ w0_{WD}, 0\ w1)\}$$

Note that the index 'WD' in the FP means that the $w0$ is performed by the same write driver as that used by the core cell that has recorded a faulty read.

As there is only one faulty read:

$$eFP_{faulty} = eFP_0$$

Now, we build the history of the remaining correct read as follow:

$$H_0' = 0\ w1\ B^6\ RES^2\ B^6\ RES^2\ B^6\ RES^2\ B^6\ r1$$

and AIV_0' related to H_0' is the following:

Ad_{i-1}				WD
I/O	column	operation	break	Transition
1	**1**	**w0**	**0**	-1

From H_0' and AIV_0' we obtain eFP_0':

$$eFP_0' = \{(0\ w1), (1\ r1), (1\ w0_{I/O,c}, 1\ r1)\}$$

As there is only one correct read, once again:

$$eFP_{fault-free} = eFP_0'$$

From *Eq. II.4*, we obtain eFP_{report} as follow:

$$eFP_{report} = \{(1\ w0_{WD}, 0\ w1)\}$$

The fault model related to eFP_{report} is a SWDF.

The same procedure is followed for the remaining syndrome. We obtain for CC_{31}:

$$eFP_{report} = \{(1\ w0_{WD}, 0\ w1)\}$$

meaning that both results lead to the same conclusion: write driver of I/O1 is affected by a SWDF.

II.2.5. Experimental results

The proposed history-based diagnosis approach has been implemented in a tool of about 7000 C++ code lines. This tool allows performing a relevant number of experiments (2000 reported in the Section). As shown in Figure II.16, the three main input data of the diagnosis tool are the *net list* (memory architecture), the *stimulus* (March Test) and the *signatures* (tester report). For the generation of the tester report, we have used a memory simulator in which we have injected a fault model for each experiment so as to mimic the behavior of an ATE. This memory simulator is an extended version of the one presented in [BEN06] that we have configured as a 512x512 SRAM with 128 I/O (write driver and sense amplifier) blocks, *i.e.* one I/O for group of four columns.

In the following sub-sections, we first compare the efficiency of a basic signature-based diagnosis approach [LI01] with the proposed history-based solution. Then, we provide additional extensive results to prove the effectiveness of the proposed diagnosis solution.

II.2.5.1. Signature vs. history-based diagnosis

In our experiments, we have considered several fault models for which we have applied the signature-based and history based diagnoses. Table II.6 presents the results given by both solutions. The first and second columns give the fault model injected (FMod) and its location (Location) in the memory. Column 3 specifies the applied test algorithm and the addressing order used during the test application (Test). The last two columns give the diagnosis report with the two approaches. For the four simulated fault injection scenarios, the developed SRAM fault simulator generates the syndromes that are used as inputs for the signature-based (SB) and history-based (HB) diagnosis.

FMod	Location	Test	SB	HB
SAF1	$CC_{99,3}$	March C-, *fast C*	**SAF1** IRF RDF	**SAF1,** IRF RDF
TF0	$CC_{123,12}$	March C-, *fast C*	**TF0**	**TF0**
dRDF (3 r1)	$CC_{99,3}$	March C-, *fast R*	SAF0 TF1 RDF IRF	SAF0, TF1 RDF, IRF **dRDF(1 r1, 6 r1)**
URWF (w1, r0)	WD $C_{0\ to\ 3}$	March C-, *fast C*	SAF1 IRF RDF	SAF1, IRF, RDF **URWF (WD: w1, r0)**

Table II.6 – Signature vs. history-based diagnosis

Let us first discuss the first two scenarios. A SAF1 has been injected in $CC_{99,3}$ (core-cell placed on line 99 and column 3) and a TF0 on $CC_{123,12}$. For the first scenario, the two diagnosis solutions return the same list of fault candidates that contains the injected fault model (SAF1). For the second scenario, a unique fault candidate is returned (TF0). Such results were rather predictable because these fault models are static.

The following two scenarios deal with dynamic fault models. In the case of dRDF in $CC_{99,3}$, the term (3 $r1$) indicates that $CC_{99,3}$ flips after three consecutive $r1$*s* following a $w1$ (see Section II.2.4.1). The employed test algorithm is again the March C-, with *fast R* (row after row) addressing order. As reported on the two last columns, the SB diagnosis reports four fault candidates (SAF0, TF1, RDF and IRF), all static, and does not include the actually injected fault model. As suspects, our solution returns not only static faults but also the dynamic fault dRDF with the reference (1 $r1$, 6 $r1$). The latter suggests that the core-cell $CC_{99,3}$ displays a faulty swap due to a dRDF, after being accessed for one to six consecutive $r1$ operations. The second injected dynamic fault model is an URWF, which requires the couple of operations ($w1, r0$) in order to be detected (see Section II.2.4.1). This fault is due to a resistive defect in the write driver (WD) used by the first four columns of the simulated SRAM ($C_{0\ to\ 3}$). The applied algorithm is again March C- with *fast C* addressing order. Compared to the other scenarios, we obtain a syndrome for each core-cell connected to the defective write driver. The SB diagnosis approach reports that each core-cell connected to the faulty write driver can be affected by SAF1, IRF or RDF. In this diagnosis response, a unique

fault model that explains the actual root cause of all syndromes is not present. The HB approach confirms the possible occurrence of SAF1, IRF and RDF, suggested by the SB approach, but it also presents the URWF as fault candidate. The URWF is the actual cause of all syndromes and it is due to a defect in the write driver sensitized by the test pattern $(w1, r0)$.

The analysis of the results in Table II.6 shows that both SB and HB methodologies are effective for the diagnosis of static faults. However, in the case of dynamic fault injection, the analysis reveals that only the HB method is able to return the correct solution, by taking in account the set of all syndromes, the memory architecture and other parameters specific of this kind of fault models. This comparative study demonstrates the interest of the proposed history-based diagnosis approach that not only extends the diagnosis to fault model difficult to track (dynamic), but which is also able to determine the faulty memory component.

II.2.5.2. Additional results

In this sub-section, we provide a larger quantity of experimental results that demonstrate the efficiency of our diagnosis approach. These results are the outcome of 1000 fault injection experiments. The faults are introduced in randomly chosen memory locations. For the case of write driver and pre-charge circuitries we performed an exhaustive set of injections.

The results of the first set of experiments are summarized in Table II.7. The first and second columns give the detail of the injected fault: fault model (Scenario) and its location (Location). The faults have been injected in the core-cells (memory array), write driver (WD) and pre-charge circuit (PRE). Column 3 specifies the applied test algorithm and the addressing order used during the test implementation (Test). The fourth column shows the average diagnosis resolution (R), which indicates the number of suspected fault primitives provided by the diagnosis tool. It also indicates if the actual injected fault is present in this set of suspects (Y: yes, N: no). For each suspected FP, the faulty memory component is associated. The last column (L) gives the percentage of correct location of the faulty memory component (the same faulty model can be related to different defective components).

Scenario	Location	Test	R	L
dRDF (1 r1)	Memory Array	March C-, *fast R*	5.8 (Y)	89%
dRDF (1 r0)	Memory Array	March C-, *fast R*	3.2 (Y)	75%
SWDF (w1, w0)	WD	March C-, *fast R*	2.25 (Y)	95%
SWDF (w0, w1)	WD	March C-, *fast R*	1.5 (Y)	95%
URWF (w1, r0)	WD	March C-, *fast C*	2, (Y)	97%
URWF (w0, r1)	WD	March C-, *fast C*	4.6 (Y)	30%
URWF (w1, r0)	PRE	March C-, *fast C*	2, (Y)	98%
URWF (w0, r1)	PRE	March C-, *fast C*	4.2 (Y)	58%
URDWF (w1, r0)	WD	March C-, *fast C*	2 (Y)	97%
URDWF (w0, r1)	WD	March C-, *fast C*	4.6 (Y)	30%

Table II.7 – Experimental results March C-

From the analysis of Table II.7, we can observe that

- The diagnosis tool always determines the root cause of the observed error (*i.e.* the injected fault) and correctly locates it (Memory Array, write driver or pre-charge circuitry).
- The achieved resolution (R) is defined as the absolute number of FPs provided by our tool. This value returns the DR if referred to the whole number of realistic FPs (about 50 for single cell FP [VDG00]). The tool always provides the correct FP, among those suspected.
- The tool always associates the correct FP with the correct defective memory component. The success of fault location is also good for the case of the remaining suspected FPs, reaching in some case a success close to 100%.

A second set of experiments have been performed using a different March Test. In this case, the test algorithm is March AB-, shown in Figure II.18 that is a modified version of the March AB [BEN05], able to cover the URDWF and URWF.

$$\{ \updownarrow (w0) \uparrow (r0, w1, r1, w1) \uparrow (r1, w0, r0, w0)$$
$$\downarrow (r0, w1, r1, w1) \downarrow (r1, w0, r0, w0) \updownarrow (r0) \}$$

Figure II.18 – March AB-

Table II.8 gives the results of these experiments exposed like in Table II.7.

Scenario	Location	Test	R	L
dRDF (1 r1)	Memory Array	March AB-, *fast R*	4.2 (Y)	92%
dRDF(1 r0)	Memory Array	March AB-, *fast R*	2.2 (Y)	77%
SWDF (w1, w0)	WD	March AB-, *fast R*	1 (Y)	100%
SWDF (w0, w1)	WD	March AB-, *fast R*	1 (Y)	100%
URWF (w1, r0)	WD	March AB-, *fast C*	2 (Y)	99.2%
URWF (w0, r1)	WD	March AB-, *fast C*	2.8 (Y)	69.5%
URWF (w1, r0)	PRE	March AB-, *fast C*	2.8 (Y)	99.2%
URWF (w0, r1)	PRE	March AB-, *fast C*	2.75 (Y)	72%
URDWF (w1, r0)	WD	March AB-, *fast C*	2 (Y)	99.2%
URDWF (w0, r1)	WD	March AB-, *fast C*	2.8 (Y)	69.5%

Table II.8 – Experimental Results March AB-

Both the parameters R and L reveal an improvement with respect to the result coming from the used of the March C-. This can be explained with the fact that March AB- returns a signature presenting more elements (read operations) than March C- (8 vs. 5), resulting in a larger information exploited during the diagnosis. This experimentation demonstrates that our diagnosis approach and the proposed tool are very efficient and provide reliable information, usable during the following phases of the failure analysis process. Moreover, the results are obtained with low hardware and time requirements.

II.2.6. Further improvements

In the previous sections, we have proven the efficiency of the proposed history-based diagnosis approach in providing accurate diagnosis reports in presence of both static and dynamic faults. This solution is also able to provide information on defect location, *i.e.* identification of the faulty memory component. However, at this stage of development, this diagnosis solution is able to deal only with single cell static faults such as SAF and TF. The first improvement that we intend to introduce will be the capability to deal with two-cell static faults such as coupling faults (CFst, CFid, CFinv) [VDG98]. This improvement will be achieved by taking in account additional information in the AIV vectors, concerning operations performed on potential aggressor core-cells.

Dynamic faults diagnosis is one of the major features of the proposed diagnosis solution. For the moment, we are able to diagnose single cell dynamic faults such as dRDF and some two-cell dynamic fault such as URWF and SWDF. As mentioned above, further dynamic fault models have to be implemented in the simulator, such as:

- ADOF (Address Decoder Open Fault)
- d2cIRF (dynamic 2-cell Incorrect Read Fault)

Among these, we consider for example the ADOF, which is a dynamic fault caused by resistive-open defects in the address decoder, having the following definition:

Address Decoder Open Fault (ADOF): *A decoder is said to have an ADOF when changing only one bit on its address results in selecting this new address but also the previous one. Consequently, two core-cells are selected at the same time for a read or a write operation.*

The ADOF requires the following test pattern to be detected:

$$wx_{CCA}\ w\bar{x}_{CCB}\ rx_{CCA}$$

where CC_A and CC_B are two core-cells, whose addresses present an Hamming distance of one. In order to make our diagnosis tool able to deal with this dynamic fault, we must take in account the parameter the Hamming distance of the addresses of the selected cell as well as the data background, (logic values stored and read during the test).

II.2.7. Conclusion

In this chapter, we have proposed a new diagnosis approach that represents a valid alternative to the signature-based approaches. This new diagnosis technique, based on the *effect-cause* paradigm, consists in creating a database containing the history record of the operations (read and write) performed on the core-cells that return incorrect logic values during the read action. In order to achieve a relevant number of experiments, we have created a dedicated software tool. Experimental results demonstrate the effectiveness of the proposed approach in returning an exhaustive set of fault candidates (static and dynamic) and providing information on the fault location.

General Conclusion

As the demand of storage capacities are constantly growing, memories are becoming SoC area dominant. Moreover, with the scaling down of such components, memories are much more sensitive to technological deviations than the standard logic. Thus, a high level of reliability for memories has to be achieved by creating and applying aggressive test procedures and then generating efficient diagnosis solutions helpful to track faulty behaviors.

Solutions currently used for SRAM testing mainly focus on the detection of static faults, but they are not able to deal with new faults called dynamic faults. These faults are mainly due to resistive-open defects and require a sequence of at least two operations (read/write) to be sensitized. The first objective of this thesis has been to study the SRAM behavior in presence of resistive-open defects involving dynamic faults and then propose effective test algorithms. Especially, this work has been focused on SRAM write drivers and sense amplifiers. Concerning the write drivers, we have shown that dynamic faults, modeled as Slow Write Driver Fault (SWDF) and Un-Restored Destructive Write Faults (URDWF) are related to some resistive-open defects in the control part of the write driver structure. We have established the conditions useful for the sensitization and the observation of these faults and we have demonstrated that a well known March algorithm called March C- is able to detect them. The second memory block we have studied is the sense amplifier. A new dynamic fault model has been proposed. It is defined as dynamic two-cell Incorrect Read Faults of two different types (d2cIRF1 and d2cIRF2). Such fault models represent failures in the sense amplifier which prevent it to do its function, *i.e.* a read operation. We have shown that the March C- with a specific addressing order and data background is able to deal with such fault model. Finally, we have highlighted that local variations of the threshold voltages (V_{TH}) impacting the core-cell functioning may be the cause of dynamic behavior, especially dynamic Read Destructive Fault (dRDF). This kind of faulty behavior is also detected by the March C- algorithm with a line after line addressing order.

The second objective of this thesis has been dedicated to memory diagnosis where two different approaches have been considered. The first one, known as Design For Diagnosis (DFD), is based on the use of extra hardware modules allowing the verification of specific SRAM blocks requirements (voltage and current levels). In this context, we have provided two solutions allowing to track weak write drivers. They are industrially viable as they require a low extra area (about 0.5% for a 512x512 SRAM). The second diagnosis approach is more general and is based on software developments useful to determine FFM and their precise localizations (core-cell, pre-charge circuit…). Classical software-based diagnosis solutions

are based on signature analysis and most of time, do not consider dynamic faults. So, a first step of this work has been to take into account these dynamic faults. As first result, a solution consisting in adding extra information (addressing order, data background) on the signature has been proposed. However, such technique, based on the *cause-effect* paradigm is limited by the *a priori* knowledge of the considered FFM. Consequently, we have proposed a new software-based diagnosis solution, called history-based diagnosis. Besides the ability of considering dynamic faults, such technique presents the major advantage to be based on the *effect-cause* paradigm, *i.e.* the *a priori* knowledge of considered FFM is not required. In order to validate this technique, a diagnosis tool has been developed and results have been provided.

Works done on this manuscript propose some memory reliability (test and diagnosis) trends. Memory test and diagnostic are really hard topics and many works are still open. As we move toward the end of the silicon roadmap, it becomes difficult to track all subtle defects. Consequently, aggressive test phases must be first developed and testing memories in different PVT corners becomes necessary. Actually, as seen in the thesis, some defects better manifest themselves at high voltage (resistive-open defects) whereas some others are more easily detected at low voltage (V_{TH} mismatches). It may be interesting to focus on other possible causes of faulty behaviors in SRAMs, such as short circuits (resistive or not), gate oxide shorts (GOS) and determine their worst case PVT corners. This study would be helpful to enlarge defect detection capabilities, and then ensuring a better memory diagnosis.

On the other hand, high memory diagnostic resolution is also required for yield ramp up. Of course, diagnosis solutions are useful to precisely point out faulty memory blocks and/or functionalities. Based on DFD modules or on software-based techniques, the aim is still the same, *i.e.* help designers and process engineers to understand memory faulty behaviors and their physical origins in order to improve memory design and/or manufacturing process. Nevertheless, these improvements present some limitations as it is not possible to manufacture a memory without any defects. So, diagnosis solutions are also helpful during the production phase when designers need to localize faulty sites in order to repair them. Consequently, it would mandatory to develop efficient diagnosis procedures able to deal with static as well as dynamic faults and also able to precisely localize faulty sites. In that context, we have already worked on a new software-based diagnosis solution. However, it should be interesting to complete it in order to take into account all new dynamic faults.

This ‘race’ to track all defects and repair faulty memories will become limited by technological advances. In fact, as said above, detecting all defects may be achieved by

multiply tests and develop efficient diagnosis procedures. However, it may be too long for industrial purposes and sometimes, certain defect will still escape test procedures. Consequently, considering all defects in a memory seems to be very complicated. We can thus imagine to classify memories according to the application requirements. Then, we can assume to embed memories with faulty behaviors that do not disturb the application.

Scientific Contributions

Publication in Journal

[TVLSI08] **A. Ney, P. Girard, C. Landrault, S. Pravossoudovitch, A. Virazel, M. Bastian,**
"Analysis of Resistive-Open Defects in SRAM Sense Amplifiers"
IEEE Transactions on Very Large Scale Integration Systems.

Publications in International Conferences Proceedings

[DATE07] **A. Ney, P. Girard, C. Landrault, S. Pravossoudovitch, A. Virazel, M. Bastian,**
"Slow Write Driver Faults in 65nm SRAM Technology: Analysis and March Test Solution"
10th IEEE Design Automation and Test in Europe, Nice, France, April 2007, pp 528-533.

[VTS07] **A. Ney, P. Girard, C. Landrault, S. Pravossoudovitch, A. Virazel, M. Bastian,**
"Un-Restored Destructive Write Faults due to Resistive-Open Defects in the Write Driver of SRAMs"
25th IEEE VLSI Test Symposium, Berkeley, USA, May 2007, pp 361-366.

[ETS07] **A. Ney, P. Girard, C. Landrault, S. Pravossoudovitch, A. Virazel, M. Bastian,**
"Dynamic Two-Cell Incorrect Read Fault due to Resistive-Open Defects in the Sense Amplifiers of SRAMs"
12th IEEE European Test Symposium, Freiburg, Germany, May 2007, pp 97-102.

[ATS07] **M. Bastian, V. Gouin, P. Girard, C. Landrault, A. Ney, S. Pravossoudovitch, A. Virazel,**
"Influence of Threshold Voltage Deviation on 90nm SRAM Core-Cell Behavior"
16th IEEE Asian Test Symposium, Beijing, China, October 2007, pp 501-504.

[DATE08] **A. Ney, P. Girard, C. Landrault, S. Pravossoudovitch, A. Virazel, M. Bastian, V. Gouin,**
"A Design-for-Diagnosis Technique for SRAM Write Drivers"

11th IEEE Design Automation and Test in Europe, Munich, Germany, March 2008, pp 1480-1485.

[DTIS08] **A. Ney, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel,**
"A Signature-based Approach for Diagnosis of Dynamic Faults in SRAMs"
3rd IEEE International Conference on Design & Test of Integrated Systems in Nanoscale Technology, Tozeur, Tunisia, March 2008, pp xxx-xxx.

[VTS08] **A. Ney, P. Girard, C. Landrault, S. Pravossoudovitch, A. Virazel, M. Bastian, V. Gouin,**
"An SRAM Design-for-Diagnosis Solution based on Write Driver Voltage Sensing"
26th IEEE VLSI Test Symposium, San Diego, USA, May 2008, pp 89-94.

[ITC08] **A. Ney, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel,**
"A History-based Diagnosis Technique for Static and Dynamic Faults in SRAMs"
To appear in Proc. of IEEE International Test Conference, Santa Clara, USA, October 2008.

Publications in national Conferences Proceedings (France)

[GDR07] **A. Ney, P. Girard, C. Landrault, S. Pravossoudovitch, A. Virazel, M. Bastian,**
"Resistive-Open Defect Influences in SRAM I/O Circuitry"
Proc. Groupement De Recherche SOC-SIP, Paris, France, June 2007.

[GDR08] **A. Ney, A. Bosio, L. Dilillo, P. Girard, C. Landrault, S. Pravossoudovitch, A. Virazel,**
"A History-Based Technique for Faults Diagnosis in SRAMs"
Proc. GDR SOC-SIP : Groupement De Recherche SOC-SIP, Paris, France, June 2008.

International Seminars

[SETS06] **A. Ney, P. Girard, S. Pravossoudovitch, A. Virazel,**

"Test of Dynamic Faults in SRAM Memories"

South European Test Symposium (SETS), Tyrol, Austria, March 2006.

[SETS07] **A. Ney, P. Girard, C. Landrault, S. Pravossoudovitch, A. Virazel,**

"Impact of threshold voltage deviation in SRAM Core-Cells"

South European Test Symposium (SETS), Sestrière, Italy, March 2007.

Submitted Article

[DATE09] **A. Ney, P. Girard, C. Landrault, S. Pravossoudovitch, A. Virazel, M. Bastian, V. Gouin,**

"Word Line Enabling Technique: a New DFT Technique for Stability Fault Testing"

Submitted in IEEE Design Automation and Test in Europe.

References

[ABR84] M. Abramovici, P.R. Menon and D.T. Miller, “Critical Path Tracing – An Alternative to Fault Simulation”, IEEE Design & Test of Computers, Vol. 1, N°1, February 1984, pp 83-92.

[ABR90] M. Abramovici, M.A. Breuer and A.D. Friedman, “Digital System Testing and Testable Design”, IEEE Press, 1990.

[ADA96] R.D. Adams and E.S. Cooley, "Analysis of a Deceptive Destructive Read Memory Fault Model and Recommended Testing", IEEE North Atlantic Test Workshop, May 1996.

[ADA97] R.D. Adams and E. S. Cooley, "False Write Through and Un-Restored Write Electrical Level Fault Models for SRAMs", Records of IEEE Int. Workshop on Memory Technology, Design and Testing, 1997, pp. 27-32.

[ADA02] R.D. Adams, “High Performance Memory Testing”, Kluwer Academic Publishers, Sept. 2002.

[APP06] D. Appello, V. Tancorre, P. Bernardi, M. Grosso, M. Rebaudengo and M. Sonza Reorda, “Embedded Memory Diagnosis: An Industrial Workflow”, Proc. of International Test Conference, 2006, pp.1-9.

[ARS01] Z. Al-Ars and A.J. van de Goor, "Static and Dynamic Behavior of Memory Cell Array Opens and Shorts in Embedded DRAMs", Proc. of Design Automation and Test in Europe, 2001, pp. 496-503.

[AZI05] M. Azimane, A. Majhi, G. Gronthoud, M. Lousberg, S. Eichenberger, and A. Ruiz, “A New Algorithm for Dynamic Faults Detection in RAMs”, Proc. IEEE VLSI Test Symposium, 2005, pp. 177-182.

[BAK99] K Baker, G Gronthoud, M Lousberg, I Schanstra and C Hawkins, “Defect-Based Delay Testing of Resistive Vias-Contacts. A Critical Evaluation”, Proc. of International Test Conference, 1999, pp. 467-476.

[BEN05a] A. Benso, A. Bosio, S. Di Carlo, G. Di Natale and P. Prinetto, “March AB, March AB1: new March tests for unlinked dynamic memory faults”, Proc. of International Test Conference, 2005.

[BEN05b] A. Benso, A. Bosio, S. Di Carlo, G. Di Natale and P. Prinetto, “Automatic March tests generation for static and dynamic faults in SRAMs”, Proc. of European Test Symposium, 2005.

[BEN06] A. Benso, A. Bosio, S. Di Carlo, G. Di Natale and P. Prinetto, “Memory Fault Simulator for Static-Linked Faults”, Proc. of Asian Test Symposium, 2006, pp. 31-36.

[BHA01] A.J. Bhavnagarwala, X. Tang and J.D. Meindl, "The Impact of Intrinsic Device Fluctuations on CMOS SRAM Cell Stability", JSSC, April 2001.

[BOR03a] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, V. De, "Parameter Variations and Impact on Circuits and Microarchitecture", Proc. of Design Automation Conference, 2003, pp.338-342.

[BOR03b] S. Borri, M. Hage-Hassan, P. Girard, S. Pravossoudovitch, A. Virazel,, "Defect-Oriented Dynamic Fault Models for Embedded SRAMs”, Proc. of European Test Workshop, 2003, pp. 23-27.

[BOR05] S. Borri, M. Hage-Hassan, L. Dilillo, P. Girard, S. Pravossoudovitch and A. Virazel, “Analysis of Dynamic Faults in Embedded-SRAMs: Implications for Memory Test”, Journal of Electronic Testing Theory and Applications, Vol. 21, No 2, April 2005, pp 169-179.

[CHA89] M.F. Chang, W.K. Fuchs and J.H. Patel, “Diagnosis and Repair of Memory with Coupling Faults”, IEEE Transactions on Computers, vol. 38, no. 4, April 1989, pp. 493-500.

[CHE05] Q. Chen, H. Mahmoodi, S. Bhunia and K. Roy, "Modeling and Testing of SRAM for New Failure Mechanisms due to Process Variations in Nanoscale CMOS", Proc. of IEEE VLSI Test Symposium, 2005, pp. 292-297.

[DEK90] R. Dekker, F. Beenker and L. Thijssen, “A Realistic Fault Model and Test Algorithms for Static Random Access Memories", IEEE Trans. on Computers, 1990, pp. 567-572.

[DIL04a] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel and S. Borri, "March iC-: An Improved Version of March C- for ADOFs Detection", Proc. of IEEE VLSI Test Symposium, 2004, pp. 129-134.

[DIL04b] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, S. Borri and M. Hage-Hassan, "Dynamic Read Destructive Faults in Embedded SRAMs: Analysis and March Test Solution", Proc. of IEEE European Test Symposium, 2004, pp 140-145.

[DIL05a] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel and M. Hage-Hassan, "Data Retention Fault in SRAMs: Analysis and Detection Procedures", Proc. of IEEE VLSI Test Symposium, 2005, pp. 183-188.

[DIL05b] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, "Resisitive open defect influence in SRAM pre-charge circuit: Analysis and characterization", Proc. of IEEE European Test Symposium, 2005, pp 116-121.

[DIL05c] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, S. Borri and M. Hage-Hassan, “Efficient March Test Procedure for Dynamic Read Destructive Fault Detection in SRAMs”, Journal of Electronic Testing Theory and Applications, Vol. 21, No 5, October 2005, pp 551-561.

[DIL06a] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, M. Bastian, "March Pre: an Efficient Test for Resistive-Open Defects in the SRAM Pre-charge Circuit", Proc. of IEEE Design and Diagnostics of Electronic Circuits and systems, 2006, pp 254-259.

[DIL06b] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, S. Borri and M. Hage-Hassan, “ADOFs and Resistive-ADOFs in SRAM Address Decoders: Test Conditions and March Solutions”, Journal of Electronic Testing Theory and Applications, Vol. 22, No 3, June 2006, pp 287-296.

[DIL07] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel and M. Bastian, “Analysis and Test of Resistive-Open Defects in SRAM Pre-Charge Circuits”, Journal of Electronic Testing Theory and Applications, Vol. 23, No 5 October 2007, pp 435-444.

[DIN00] D.M. Kwai, H.W. Chang, H.J. Liao, C.H. Chiao and Y.F. Chou, “Detection of SRAM Cell Stability by Lowering Array Supply Voltage”, Proc. of IEEE Asian Test Symposium, 2000, pp 268-273.

[HAM02] S. Hamdioui, Z Al-Ars and A.J. van de Goor, "Testing Static and Dynamic Faults in Random Access Memories", Proc. IEEE VLSI Test Symposium, 2002, pp. 395-400.

[HAM03] S. Hamdioui, R. Wadsworth, J.D. Reyes and A.J. van de Goor, "Importance of Dynamic Faults for New SRAM Technologies", Proc. of IEEE European Test Workshop, 2003, pp. 29-34.

[HAR01] T.P. Haraszti, “CMOS Memory Circuits”, Kluwer Academics Publishers (second edition), 2001, pp 402.

[HAR07] S.M. Al-Harbi, F. Noor, F.M. Al-Turjman, “March DSS: A New Diagnostic March Test for All Memory Simple Static Faults”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 26, N° 9, September 2007, pp. 1713-1720.

[JAM01] C.M. James, C.W. Tseng and E.J. McCluskey, “Testing for Resistive Opens and Stuck Opens”, Proc. of International Test Conference, 2001, pp. 1049-1058.

[KES01] A. Keshavarzi, S. Ma, S. Narendra, B. Bloechel, K. Mistry, T. Ghani, S. Borkar, V. De, “Effectiveness of Reverse Body Bias for Leakage Control in Scaled dual Vt CMOS ICs”, IEEE Proc. International Symposium on Low Power Electronics and Design, Aug. 2001, pp. 207-212.

[KIM98] I. Kim, Y. Zorian, G. Komoriya, H. Pham, F.P. Higgins and J.L. Lewandowski, “Built in Self Repair for Embedded high Density SRAM”, Proc. of International Test Conference, pp 1112-1119, 1998.

[LI01] J.-F. Li, K.-L. Cheng, C.-T. Huang and C.-W. Wu, “March-Based RAM Diagnosis Algorithms for Stuck-At and Coupling Faults”, Proc. of International Test Conference, 2001, pp. 758-767.

[MAL96] W. Maly, H. Heineken, J. Khare and P.K. Nag, “Design for Manufacturability in submicron domain”, Proc. of ICCAD 96, Nov. 96, pp 690-697.

[MEI97] A. Meixner and Jash Banik, “Weak Write Test Mode: An SRAM Cell Stability Design For Test Technique”, Proc. of IEEE International Test Conference, 2001, pp. 1043-1052.

[NICO96] M. Nicolaidis, “Theory of Transparent BIST for RAMs”, IEEE Transaction On Computers, vol. 45, N° 10, October 1996, pp. 1141-1155.

[NIG98] D. Niggemeyer, M. Redeker and J. Otterstedt, “Integration of Non-classical Faults in Standard March Tests”, Records of the International Workshop on Memory Technology, Design and Testing, 1998.

[NIG00] D. Niggemeyer, M. Redeker and E.M. Rudnick, “Diagnostic testing of embedded memories based on output tracing”, Proc. of Memory Technology, Design and Testing, 2000, pp. 113-118.

[PAV04] A. Pavlov, M. Sachdev and J. Pineda de Gyvez, “An SRAM Weak Cell Fault Model and a DFT Technique With a Programmable Detection Threshold”, Proc. of IEEE International Test Conference, 2004, pp. 1006-1015.

[PAV05] A. Pavlov, M. Azimane, J. Pineda de Gyvez and M. Sachdev, “Word Line Pulsing Technique for Stability Fault Detection in SRAM Cells”, Proc. of IEEE International Test Conference, 2005, paper 33.1.

[PAV06] A. Pavlov, M. Sachdev and J. Pineda de Gyvez, “Weak Cell Detection in Deep-Submicron SRAMs: A Programmable Detection Technique”, IEEE Journal of Solid-State Circuits, Vol. 41, Issue 10, Oct. 2006, pp. 2334-2343.

[PIL01] H. Pilo, R. Dean Adams, R.E. Busch, E.A. Nelson and G.E. Rudgers, “Bitline Contacts in High Density SRAMs: Design for Testability and Stressability”, Proc. of International Test Conference, 2001, pp 776-782.

[RON02] E. Rondey, Y. Tellier, S. Borri, “A Silicon-Based Yield Gain Evaluation Methodology for Embedded SRAMs with Different Redundancy Scenarios”, Proc. of Memory Technology, Design and Testing, 2002, pp. 57-61.

[SAC97] M. Sachdev, “Open Defects in CMOS RAM Address Decoders”, IEEE Design & Test of Computers, Vol.14, N°2, April-June 1997, pp. 26-33.

[SUK81] D.S. Suk. and S.M. Reddy, “A March Test for Functional Faults in Semiconductor Random Access Memories”, IEEE Transaction on Computers, 1981, pp 982-985.

[THA06] S.K. Thakur, R. Parekhji and A.N. Chandorkar, “On-chip Test and Repair of Memories for static and Dynamic faults”, Proc. of International Test Conference, 2006.

[TER98] C. Terwiesch and R.E. Bohn, “Learning and process improvement during Production Ramp up”, IEEE Int. Journal of Production Economics, Vol. 70, n° 1, 1998, pp 1-19.

[VDG98] A.J. van de Goor, “Testing Semiconductor Memories, Theory and Practice”, COMTEX Publishing, Gouda, The Netherlands, 1998.

[VDG99] A.J. van de Goor and J.E. Simonse, “Defining SRAM Resistive Defects and Their Simulation Stimuli," Proc. of Asian Test Symposium, 1999, pp. 33-40.

[VDG00] A.J. van de Goor and Z. Al-Ars, "Functional Memory Faults: A Formal Notation and a Taxonomy", Proc. of IEEE VLSI Test Symposium, 2000, pp. 281-289.

[VDG04] A.J. van de Goor, S. Hamdioui and R. Wadsworth, "Detecting Faults in the Peripheral Circuits and an Evaluation of SRAM Tests", Proc. of IEEE International Test Conference, 2004, pp. 114-123.

[VAR06] V.A. Vardanian, G. Harutunyan, Y. Zorian, “Minimal March-Based Fault Location Algorithm with Partial Diagnosis for All Static Faults in Random Access Memories”, Proc. of Design and Diagnostics of Electronic Circuits and systems, 2006, pp. 260-265.

[YAR96] V.N. Yarmolik, Y.V. Klimets, A.J. van de Goor and S.N. Demidenko, “RAM diagnostic tests”, Proc. of Memory Technology, Design and Testing, 1996, pp. 100-102.

[ZAR00] K. Zarrineh, R. Dean Adams and A.P Deo, “Defect Analysis and Realistic Fault Model Extensions for Static Random Access Memories”, Records IEEE Int. Workshop on Memory, Technology, Design and Testing, 2000, pp. 119-124.

[ZOR02] Y. Zorian and S. Shoukourian, “Embedded Memory Test & Repair: Infrastructure IP for SoC Yield”, Proc. of International Test Conference, 2002, pp 340-349.

List of Figures

List of Tables

Test et Diagnostic de Fautes Dynamiques dans les Mémoires SRAM

RESUME : De nos jours, les mémoires sont présentes dans de nombreux circuits intégrés conçus pour des applications électroniques embarquées et occupent une majeure partie de la surface des systèmes sur puce (SoC). Ces mémoires deviennent donc les acteurs principaux du rendement de production. Or, une forte densité d'intégration associée à une complexité élevée des procédés de fabrications rendent ces mémoires toujours plus sensibles aux défauts de fabrications. Afin de mettre en évidence les défaillances survenant dans les mémoires, plusieurs méthodes de test existent. Ces solutions de test couramment utilisées pour les mémoires SRAM sont basées sur la détection de fautes statiques telles que les fautes de collage ou de couplage. Des algorithmes spécifiques, appelés algorithmes March, sont utilisés afin de mettre en évidence ce type de fautes. Cependant, ces solutions de test ne sont pas adaptées à la détection d'un nouveau type de faute apparaissant dans les technologies submicroniques. Ces fautes, appelées fautes dynamiques, sont principalement dues à des défauts de type « ouverts-résistif » et ne se manifestent que dans des configurations très spécifiques. En effet, une séquence d'opérations est nécessaire à la mise en évidence de ces fautes. Le premier objectif de cette thèse a été de proposer des solutions de test permettant la détection de fautes dynamiques dues à des défauts « ouverts-résistifs » dans le driver d'écriture et l'amplificateur de lecture. Une extension sur l'étude des comportements dynamiques face à des variations de procédés de fabrication dans le point mémoire a été proposée. Enfin, la seconde partie de cette thèse fournit de nouvelles solutions de diagnostic, capables de prendre en compte les fautes dynamiques d'une part, et proposant une détection précise des sites fautifs. Ces travaux ont été réalisés en collaboration avec la société Infineon basée à Sophia Antipolis spécialisée dans la conception de mémoires SRAM.

Test and Diagnostic of Dynamic Faults in SRAM memories

ABSTRACT: Nowadays, embedded memories occupy a large part of the System-on-Chip (SoC) silicon area. Consequently, memories are the main responsible for the overall System-on-Chip yield. However, a high integration density and the complexity of the fabrication process make memories more and more prone to manufacturing defects. Therefore, efficient test and diagnostic solutions for memories are required. Current test solutions used for SRAM memories are oriented to static fault detection. Recent researches show that VDSM (Very Deep SubMicron) technologies more frequently involve dynamic faults. These faults, mainly due to bad vias or contacts involving a resistive-path, need a specific pattern to be sensitized. However, classical test solutions are not able to deal with such behaviors. Consequently, the first part of this thesis is dedicated to new test solutions allowing to detect dynamic faults due to resistive-open defects in the memory. Especially, we focus our study on the write driver and the sense amplifier. New fault models and March test solutions are proposed. Then, an extension on dynamic faults is provided: a brief study on the impact of the threshold voltage variation is given. Finally, the next part of this thesis is oriented toward memory diagnostic. New efficient algorithmic diagnosis solutions are proposed. They allow dealing with dynamic faults and providing information on the faulty bloc location. This thesis has been done in the framework of the Associate MEDEA project in cooperation with Infineon Technologies.

MOTS-CLES : Mémoires SRAM, Test, Diagnostic, Conception en vue du diagnostic, Fautes Dynamiques

---

Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier, LIRMM, 161 rue Ada, 34392 Montpellier Cedex 5, France.