Alberto Bosio
Luigi Dilillo
Patrick Girard
Serge Pravossoudovitch
Arnaud Virazel

Advanced Test Methods for SRAMs

Effective Solutions for Dynamic Fault Detection in Nanoscaled Technologies

Springer

Advanced Test Methods for SRAMs

Alberto Bosio · Luigi Dilillo · Patrick Girard ·
Serge Pravossoudovitch · Arnaud Virazel

Advanced Test Methods for SRAMs

Effective Solutions for Dynamic Fault Detection in Nanoscaled Technologies

Springer

Alberto Bosio
LIRMM - Laboratoire d'Informatique de Robotique de Microélectronique de Montpellier
161 Rue Ada
34392 Montpellier
France
alberto.bosio@lirmm.fr

Luigi Dilillo
LIRMM - Laboratoire d'Informatique de Robotique de Microélectronique de Montpellier
161 Rue Ada
34392 Montpellier
France
luigi.dilillo@lirmm.fr

Patrick Girard
LIRMM - Laboratoire d'Informatique de Robotique de Microélectronique de Montpellier
161 Rue Ada
34392 Montpellier
France
patrick.girard@lirmm.fr

Serge Pravossoudovitch
LIRMM - Laboratoire d'Informatique de Robotique de Microélectronique de Montpellier
161 Rue Ada
34392 Montpellier
France
serge.pravossoudovitch@lirmm.fr

Arnaud Virazel
LIRMM - Laboratoire d'Informatique de Robotique de Microélectronique de Montpellier
161 Rue Ada
34392 Montpellier
France
arnaud.virazel@lirmm.fr

ISBN 978-1-4419-0937-4 e-ISBN 978-1-4419-0938-1
DOI 10.1007/978-1-4419-0938-1
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2009935341

Acknowledgements

The authors would like to acknowledge Stefano Di Carlo from Politecnico di Torino, Italy, for his meaningful help and suggestions after reading part of the first draft of the book. Many thanks go to Alexandre Ney who did his PhD on memory testing and diagnosis at LIRMM and provided part of the material reported in this book. The authors also wish to thank Jean-Christophe Vial, Magali Bastian, Simone Borri, Vincent Gouin, and Nabil Badereddine, from Infineon Technologies, Sophia Antipolis, France, for the long-term and fruitful cooperation with the authors in the field of SRAM memory testing.

Advanced Test Solutions for Dynamic Faults in SRAM Memories

Authors of the book:

Alberto Bosio, Associate Professor, LIRMM/University of Montpellier – France
Luigi Dilillo, CNRS Researcher, LIRMM/CNRS – France
Patrick Girard, CNRS Research Director, LIRMM/CNRS – France
Serge Pravossoudovitch, Professor, LIRMM/University of Montpellier – France
Arnaud Virazel, Associate Professor, LIRMM/University of Montpellier – France

Summary and objective of the book:

Memory design and test represent very important issues. Memories are designed to exploit the technology limits to reach the highest storage density and high speed access. The main consequence is that memory devices are statistically more likely to be affected by manufacturing defects. The challenge of testing SRAM memories consists in providing realistic fault models and test solutions with minimal application time. Due to the complexity of the memory device, fault modeling is not trivial. Classical memory test solutions cover the so-called 'static faults' (such as stuck-at, transition, and coupling faults) but are not sufficient to cover faults that have emerged in latest VDSM technologies and which are referred to as 'dynamic faults'.

This book represents a valuable introduction and guide to new test approaches developed so far for dealing with dynamic faults in the latest generation of SRAM memories. The book is organized in a bottom-up approach, starting from the study of the electrical causes of the malfunctions up to the generation of smart test strategies and dedicated algorithms. Based on this scheme, an exhaustive analysis of resistive-open defects is first performed on each memory component (core-cells, address decoders, write drivers, sense amplifiers, etc.) and the resulting dynamic fault models and related test solutions are enumerated. Diagnosis of dynamic faults, as well as process variations and leakage current in SRAMs, are also addressed at the end of the book.

Contents

General Introduction

History

After the era of memory technology allowing a capacity of a few bytes (in the early 1940s), magnetic-core memory became the dominant form of memory until the invention of the transistor in the late 1970s. The memory technology has evolved successfully since then, following the trend characterized by the Moore's law.

Nowadays, semiconductor memories are widely used to store programs and huge volume of data in almost every digital system. For many current applications (audio, video, data processing), the system performance is strictly related to the number, capacity (size), and speed of its embedded memory units. A steadily increasing percentage of area in *Integrated Circuits* (ICs) and *Systems-on-Chip* (SoCs) is thus dedicated to implement memory components. According to the *International Technology Roadmap for Semiconductors* (ITRS) drawn by the *Semiconductor Industry Association* (SIA), memories occupied 20% of an SOC area in 1999, 52% in 2002, and are forecasted to occupy up to 94% of a typical SoC area by 2014 (ITRS 007).

As memories represent a significant part of typical SoCs, any improvement in the design and manufacturing processes of memory devices has a straightforward and significant impact on numerous features such as cost, yield, performance, and reliability of the whole SoC. For this reason, memories have historically been designed to exploit the technology limits and hence achieve the highest storage density and access speed. The main consequence is that memory devices are statistically more likely to be affected by manufacturing defects and process variations, and thus are becoming the main detractor of the overall SoC yield. The development of efficient test solutions for these devices, though representing a difficult and costly task, is therefore mandatory. Moreover, because memories are used as test vehicles for monitoring the manufacturing process and improving its yield, extracting additional diagnostic data to determine the causes of failures is also required in the testing strategy.

SRAMs

With the rapid growth in the requirement for semiconductor memories, there have been a number of technologies and types of memory that have emerged, each one having its own field of applications. A detailed description of all types of semiconductor memories is provided in Chapter 1 of this book. Among these various types of memories, *Static Random Access Memories* (SRAMs) are widely used in embedded and high-speed applications such as cache memories for processors. SRAMs are volatile memories that offer low-power consumption and in which any storage location (core-cell) can be randomly accessed in the same amount of time. SRAMs cannot retain data without power but permit data to be read from or written into the storage core-cells. The term 'static' refers to the structure of the storage cell which is typically made of six transistors in current technologies.

Test of SRAMs

As any other type of memory device, SRAMs are subject to defects or variations during the manufacturing process, and hence have to be tested efficiently. The challenge of testing SRAMs consists in providing realistic fault models and practical test solutions with minimal application time. Classical SRAM test solutions cover the so-called 'static faults', such as stuck-at, transition, and coupling faults, which require at most one read or write operation to be sensitized, and are based on specific tests called 'March' tests. A number of books have been published so far to present and discuss the taxonomy of static fault models and dedicated March test solutions (Sharma 1997, Van de Goor 1998, Adams 2002, Hamdioui 2004). However, these test solutions are not sufficient to cover a new type of faults that are emerging in the latest nanometer technologies and that are a consequence of the shrinking geometries. These new faults, referred to as 'dynamic faults' and introduced in Van de Goor and Al-Ars (2000), are sensitized by more than one read or write operations and hence require elaborate combinations of operations, data background, and addressing sequences to be detected. Typically, these faults model electrical malfunctions due to resistive defects (opens and bridges) or parameter mismatches, and may occur in the various blocs of an SRAM such as the core-cell array, the address decoders, the pre-charge circuits, the sense amplifiers, and the write drivers.

Organization of the Book

This book presents a comprehensive state-of-the-art of fault models and dedicated test and diagnostic solutions for the new class of dynamic faults. It is complementary to the above-mentioned books dedicated to static faults. The book is organized in a uniform manner for each chapter and presented with a bottom-up approach – from the transistor level up to the logic and algorithmic levels. For each bloc of an

SRAM, a structural description and the corresponding fault-free electrical behavior are first presented. Then, a detailed electrical analysis of resistive defects inducing a faulty behavior is reported. The next step consists in a functional fault modeling of resistive defects leading to dynamic faults. Finally, dedicated March-based test strategies with the corresponding complexity evaluation are presented. Diagnosis of dynamic faults, as well as the impact of process variations and leakage current on the SRAM behavior, are surveyed at the end of the book.

Description of Each Chapter

The book is composed of eight chapters. Chapter 1 presents the context of this book by first proposing an overview of semiconductor memories and then giving background on SRAM modeling and testing. Basics on March tests, functional fault modeling, automatic test generation, and test validation are also given is this chapter.

Chapter 2 up to Chapter 6 are structured in a uniform manner and address dynamic faults in various blocs of an SRAM; Chapter 2 considers resistive-open defects in core-cells, Chapter 3 considers resistive-open defects in pre-charge circuits, Chapter 4 considers resistive-open defects in address decoders, Chapter 5 considers resistive-open defects in write drivers, and Chapter 6 considers resistive-open defects in sense amplifiers. Electrical analyses reported in each chapter come from experiments performed on industrial and up-to-date SRAM technologies. Functional fault modeling of resistive defects and generation of dedicated March test algorithms are presented in each chapter.

As the silicon industry moves towards the end of the technology roadmap, controlling the manufacturing process of scaled devices is becoming a great challenge. In nanometer technologies, global (inter-die) and local (intra-die) device parameter variations are expected to occur more and more often, especially in minimum geometry transistors commonly used in area-constrained circuits such as memories (Borkar et al. 2003). Chapter 7 reports studies on dynamic faults due to process variations (called mismatches in case of local variations) in SRAMs. Impact on the threshold voltage of transistors in core-cells, on the leakage current, and on the memory read opeations are studied in this chapter.

Diagnosis can be used to improve memory yield by using redundancies and hence repair defective parts of a memory. Diagnosis can also be used to improve yield ramp up for new technologies and new designs. Here, the crucial information is not only the fault localization but also the fault type and its physical origin. With such information available, engineers can adjust the manufacturing process and/or enhance the memory design. Chapter 8 surveyed state-of-the-art diagnosis solutions for dynamic faults in SRAMs. Signature-based diagnosis, history-based diagnosis, and design-for-diagnosis of dynamic faults in write drivers are addressed in this last chapter of the book.

Chapter 1
Basics on SRAM Testing

1.1 Overview of Semiconductor Memories

Every computer system, such as super computers or small systems for basic applications, requires memory in order to store data and program instructions. Semiconductor memories differ from the other storage mechanisms such as tapes, magnetic discs, and optical discs by the access mechanism of stored data. Semiconductor memories are characterized by the fact that the stored data can be accessed in any order in a constant time, regardless of its physical location (random access).

Based on this characteristic, semiconductor memories can be classified according to the type of data storage and data access mechanisms in three main families: the Read-Write Memories (RWMs), Read-Only Memories (ROMs), and Non-Volatile Read-Write Memories (NVRWMs) (Prince 1996). Figure 1.1 gives a schematic view of the various types of memories in each family.

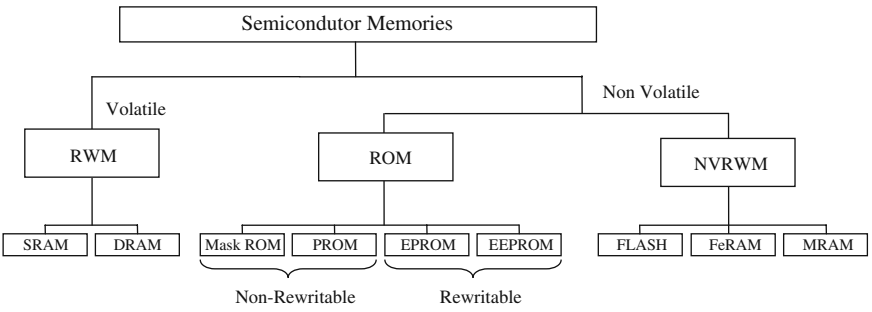


Fig. 1.1 Memory taxonomy

RWMs are volatile memories permitting data to be stored and retrieved at comparable speeds. Computer systems commonly use RWMs for data and program storage. Within this family, the time required for storing (writing) information and retrieving (reading) information is independent of the physical location of the data within the memory. Among RWM devices, Static Random Access Memories and

A. Bosio et al., *Advanced Test Methods for SRAMs*,
DOI 10.1007/978-1-4419-0938-1_1, © Springer Science+Business Media, LLC 2010

Dynamic Random Access Memories, denoted as SRAMs and DRAMs respectively, are the most popular. The primary difference between them is the lifetime of the data they store. An SRAM retains its content as long as the power supply of the memory is turned on. If the power supply is temporarily or permanently turned off, its content is lost forever. Compared to an SRAM, a DRAM is smaller, since each memory cell is commonly made with one transistor and one capacitance. On the other hand, it requires a refreshment process to keep the data value. This process makes the memory slower and more power consuming with respect to an SRAM. As for SRAM, the DRAM content is lost when the power supply is turned off. Generally, an SRAM is used in systems where a high access speed is required. On the other hand, a lower cost-per-byte makes a DRAM more attractive whenever a large amount of storage is required. Many embedded systems include both types of RAMs: a small block of SRAM (few kilobytes) along a critical data path and a much larger block of DRAM for everything else.

ROMs are non-volatile memories. Actually, a common feature of all ROM devices is their ability to retain data and programs forever, even if the power supply is turned off. A ROM allows reading the stored information at the same speed as that of a RWM, but it restricts the write operation. ROMs can be used to store a microprocessor operating system program. For example, they are also employed in operations that require look-up table, such as finding the resulting values of mathematical functions. A popular application of ROMs has been their use in video game cartridges.

ROMs are distinguished by the method used to write (program) data. The first type of ROM refers to hardwired devices that contain a preprogrammed set of data. Therefore, the content of the ROM has to be specified before chip production, so the actual data must be used to arrange the transistors inside the chip. Hardwired memories are called masked ROMs. The advantage of a masked ROM is its low production cost. The second type of ROM is the Programmable ROM (PROM), which is produced and sold in an un-programmed state. The process of writing (programming) data to the PROM requires a device programmer. The device programmer writes data to the device one word at a time by applying an electrical charge to the input pins of the chip. Once a PROM has been programmed in this way, its content can no longer be changed. If the code or data stored in the PROM must be changed, the current device must be discarded. The third type of ROM is the Erasable and Programmable ROM (EPROM). In the EPROM devices, data can be erased and rewritten. To erase data in the EPROMs, it is necessary to expose the device to a source of ultraviolet light. The last type of ROM is the Electrically Erasable and Programmable ROM (EEPROM). In the EEPROM devices, data can be erased and rewritten as for the EPROMs. The erase operation is accomplished electrically, rather than by exposure to ultraviolet light. Usually, a device programmer or special voltage is required to erase and to write data (program). Once written, the new data will remain in the device until it is electrically erased, even if the power supply is turned off.

Besides RWM and ROM, a third family of memories has emerged that combines the main features of both RWMs and ROMs. These devices can be collectively

referred to as Non-Volatile Read-Write Memories (NVRWMs). In the NVRWM, data can be read and written as desired, like RWMs, but NVRWMs maintain their content even without power supply, like ROMs. Flash memories combine the best features of the memory devices described so far. Flash memory devices are high density, low cost, non-volatile, fast (to read, but not to write), and electrically reprogrammable devices. These advantages are overwhelming and, as a direct result, the use of flash memories has increased dramatically in embedded systems. Flash devices can only be erased one sector at a time (i.e., per page), not byte-per-byte. Typical sector sizes are in the range of 256 bytes to 16 KB.

Nowadays, all kinds of memories involve some compromises. SRAMs are low power and very fast, but more expensive than DRAMs and harder to make in large sizes. DRAMs need more power and are slower than SRAMs, but much smaller in terms of size. Flash memories do not lose data when the power supply is turned off, but they are not small and very slow to store data. Therefore, despite the big success of the computer memory industry, the search always goes on for a successor technology that can compensate for some of the downsides of existing memory types while having advantages of its own. One long-term contender is the Magnetic Random Access Memories (MRAMs), which has been brewing in the labs for around 30 years (Prince 2002). With MRAMs, as with tapes or disks, the signal sets the magnetic state of the memory bit in a permanent way indefinitely, without the need to maintain power supply. Another very promising technology for non-volatile memories is the Ferroelectric Random Access Memories (FeRAMs) technology (Prince 2002), in which the polarization of a ferroelectric thin film is used for information storage.

Among the different types of semiconductor memories described above, this book deals with SRAM and addresses the test issues related to the latest SRAM technologies.

1.2 Typical Structure of an SRAM

In any kind of SRAM, the bits are either individually addressable (bit-oriented memories) or addressable in groups of 4, 8, 16, or more (word-oriented memories). For simplicity in the rest of the book, all the bits of SRAMs are considered as individually addressable (bit-oriented SRAMs). Moreover, all test solutions discussed in this book can be easily adapted to the word-oriented SRAMs according to the translating algorithm proposed in Van de Goor and Tlili (2003).

The core of the memory consists of cells in which bits are stored. Each memory cell is an electronic circuit capable of storing (at least) one bit. The data in a conventional six-transistor SRAM is stored by two stable states of a latch composed of two CMOS inverters in a feedback loop (Prince 1996). These two inverters are usually drawn as cross-coupled inverters. The complete six-transistor cell is shown in Fig. 1.2. Basically, it is composed of additional access transistors which allow the data stored in the latch to be read or a new data to be written.

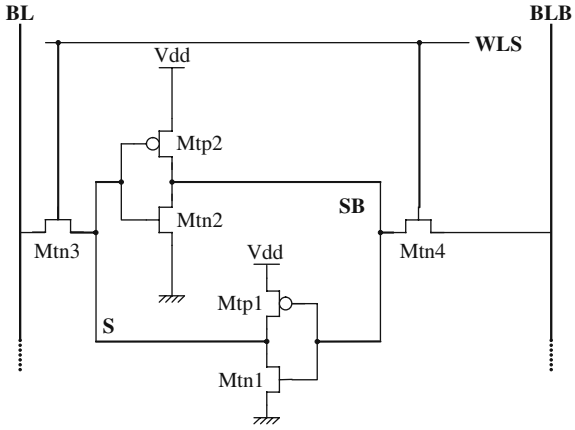


Fig. 1.2 Six-transistor SRAM core-cell

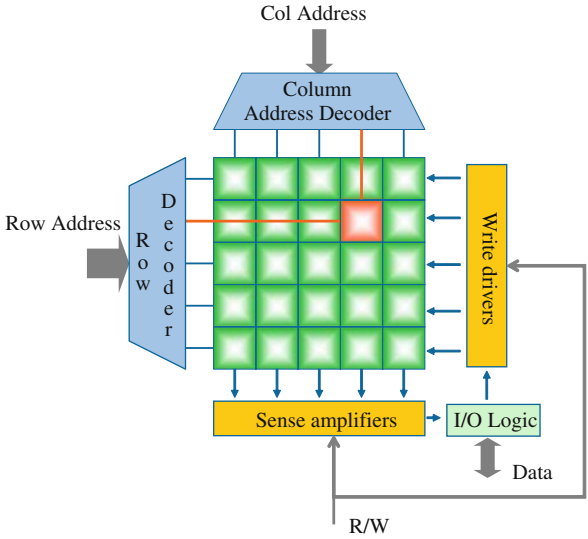


Fig. 1.3 SRAM architecture

The organization of the storage cells is commonly done in a square or nearly square matrix. Figure 1.3 illustrates such an organization. The cell matrix has 2^{MR} rows and 2^{NC} columns, for a total storage capacity of 2^{MR+NC} bits, where MR and NC are the number of bits used to specify the row address and the column address, respectively. For example, a 1 M-bit square matrix would have 1,024 rows and 1,024 columns (MR = NC = 10 bits). Each cell in the array is connected to one of the 2^{MR} row lines, universally called word lines, and to one of the 2^{NC} column lines, commonly called bit lines, or also digit lines. A particular cell can be accessed for a read or write operation by selecting its word line and its bit line. All the details about the SRAM core-cell and its functionalities are given in Chapter 2.

The activation of one of the 2^{MR} word lines is performed by the row decoder, which is a combinational logic circuit that selects the word line corresponding to the address (composed of MR bits) applied to the decoder input. When the *k*th word line is activated, whatever the operation, all the 2^{NC} cells in the *k*th row are connected to their respective bit lines. The address decoder is detailed in Chapter 4.

The R/W signal specifies the type of operation to be performed on the selected cell (Read, Write). In the case of a read operation, the sense amplifiers allow to amplify the voltage of the signal coming from the cell. In this way, the data is managed by the I/O logic that is connected to the data bus. In the case of a write operation, the write drivers act on the selected memory cell to write the data coming from the data bus. All the details about the I/O logic, write driver, and sense amplifier are given in Chapters 4, 5, and 6, respectively.

In this section, a typical SRAM structure has been presented. In the rest of this book, each chapter will target a specific SRAM block by first giving the details about how the block works, and then by analyzing the resistive-open defects affecting the block. Finally, each chapter will present efficient test solutions targeting the considered SRAM block.

1.3 The Context of SRAM Testing

For several current applications (audio, video, data processing), the system performances are strictly related to the capacity (size) and speed of its memory elements. This is one of the main reasons of the memory predominance in embedded systems and System-on-a-Chip (SoC). This is confirmed by the Semiconductor Industry Association roadmap, which forecasts that in 2014 more than 94% of the overall SoC area will be composed of memories (ITRS 2007). This is illustrated in Fig. 1.4.

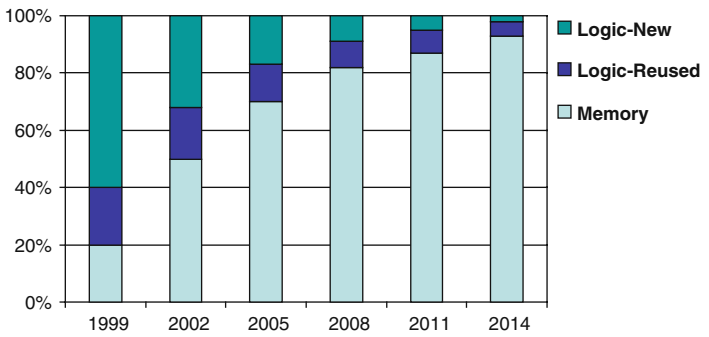


Fig. 1.4 ITRS roadmap: International Technology Roadmap for Semiconductors

Moreover, memories are designed to exploit the technology limits in order to reach the highest storage density and the highest access speed. The main consequence is that memory devices are statistically more likely to be affected by

manufacturing defects. The importance of manufacturing test is also due to its impact on the final product cost. Since memories are becoming the main responsible of the SoC yield and represent a large percentage of the product cost, searching for efficient test solutions is mandatory for these devices.

A comprehensive SRAM test must guarantee the correct functionality of each cell of the memory (ability to store and to maintain a data) and the corresponding addressing, write, and read operations in the worst conditions with regard to temperature interval, voltage constraints, and timing requirements. It must also ensure pattern and voltage sensitivity immunity, where the voltage sensitivity is a bad behavior caused by relevant variations in the voltage supply.

Functional test is the most popular type of test applied to an SRAM (Van de Goor 1998). Basically it consists in applying a set of operations (read and write) to the SRAM under test in order to highlight possible differences with respect to the expected good behavior. Functional test stems from the definition of Fault Models. Faults have physical causes, called *defects*. Defects often consist of an impurity such as a dust particle. An *error* is the observable manifestation of a defect. Such defects at the layout level of the chip are translated into electrical faults and, thereafter, are translated into logical faults such that they can be tested with logical signals ('0' and '1'). This mapping of defects into electrical, and thereafter into logical faults is called *fault modeling.* The fault modeling is related to the considered SRAM structure since a fault can affect one or more of the blocks composing an SRAM, such as the core-cell, the address decoder, the sense amplifiers, etc.

Due to the importance of fault modeling to exploit an efficient functional test, the memory model as well as the adopted fault model formalism are introduced in the following sections. Based on those formalisms, the state-of-the-art in terms of fault models is given. Finally, the state-of-the-art of functional test solutions, the so-called March tests (Van de Goor 1998), is presented.

1.3.1 Memory Model

This section introduces the formal model adopted to represent the memory under test. This model has been introduced in Niggemeyer and Rudnick (2004) and Benso et al. (2008).

Definition 1.1 An N-cell one-bit memory can be formally defined as a 5-tuple: $<D, N, M, A, OP>$

where

- $D = \{0, 1, -\}$ is the set of possible states of a memory cell;
- N is the number of cells, also called the 'size' of the memory;
- $M = (c_0, c_1, c_2, \ldots, c_{N-1}) \mid c_i \in D, 0 \leq i \leq N-1$ is the array of N cells;
- $A = (0, 1, 2, \ldots, N-1) \mid 0 \leq i \leq N-1$ is the address of the N cells;
- $OP = \{r, w_d \mid d \in D\}$ is the set of possible memory operations, where r means a read operation whereas w_d means a write operation of the value d.

As mentioned earlier only bit-oriented SRAMs are considered in this book, but the presented test solutions can be easily extended to word-oriented memories (Van de Goor and Tlili 2003).

1.3.2 Fault Model Representation

The formalism used to represent functional faults of a memory is the Fault Primitive (FP) formalism introduced in Van de Goor and Al-Ars (2000) and reminded in this section. Memory faults are modeled starting from Faulty Behaviors (FB) defined as the deviations of the memory behavior (faulty memory) from the expected one (good memory). A FB is sensitized by the application of a sequence of stimuli on the memory cells involved in the faulty behavior. The set of those memory cells is called f-cells. Based on the f-cells cardinality (| f-cells |), faults can be clustered into the following classes:

- Single-cell faults (| f-cells | = 1).
- Two-cell faults (| f-cells | = 2). In this case, it is possible to distinguish between an aggressor cell (a) and a victim cell (v). The former is the cell that sensitizes the FB and the latter is the cell that exhibits the effect of the FB.
- Multiple cell faults (| f-cells | > 2).

A stimulus S is composed of a sequence of operations represented by op_1,op_2,. . .,op_m, where op_i is defined as follows:

$$op_i = n(dO) \tag{1.1}$$

where

- $n \in A$ identifies the address of one of the f-cells;
- $d \in D$ is the initial value of the cell;
- $O \in OP \cup \{r_d \mid d \in D\}$ where OP is defined in Definition 1.1 and r_d is a read and verify operation. It means 'read the content of the cell and verify that its value is equal to d.' The operation O can be omitted when a FB is sensitized just by the f-cells being in a certain condition (e.g., state fault (Van de Goor and Al-Ars 2000)).

Examples of possible stimuli are

- S = $i(0)$ corresponds to a FB sensitized by the state of the faulty cell i equal to 0;
- S = $i(-w_1), j(0)$ corresponds to a FB sensitized by writing 1 into cell i, regardless of the current state of the cell, and by setting cell j to the logical state 0;
- S = $i(0w_1), i(1r_1)$ corresponds to a FB sensitized by a write operation of the value 1 on cell i, initially containing a 0, immediately followed by a read operation on the same cell. In this case of FB, it is possible to simplify the notation by merging

the operations performed on the same memory cell. Thus, the notation $i(0w_1)$, $i(1r_1)$ can be replaced by $i(0w_1r_1)$.

The FB is expressed using the following notation:

Definition 1.2 FB $= n(F)/n(R)$ where F represents a value $d \in D$ (Definition 1.1) set in the faulty memory cell. n and R represents the output value $d \in D$ obtained while reading the content of cell n during fault sensitization.

For example, FB $= i(0)/i(1)$ means that the faulty cell i is set to the value 0 and a read operation on the same cell i returns the value 1.

Definition 1.3 A *Fault Primitive* (FP) represents the difference between an expected (fault-free) and the observed (faulty) memory behaviors under a set of performed operations, denoted by FP = <S/FB>.

S represents the applied stimulus and FB represents the faulty behavior sensitized by S. A Functional memory Fault Model (FFM) consists of a non-empty set of FPs (Van de Goor and Al-Ars 2000). For example, a subset of the Idempotent Coupling Fault (a rising transition performed on an aggressor cell a sets the logic value of the victim cell v to 1 (Van de Goor 1998)) can be described by the following FP:

$$\mathrm{FP_{CFid}} = <a(0w_1), v(0)/v(1)/v(-)> \tag{1.2}$$

In this example, S is equal to $a(0w_1)$, $v(0)$ meaning that to sensitize the fault, a write 1 operation is required on the aggressor cell a initialized to 0, and that the victim cell v is set to 0. The FB of this fault sets the logical value of the victim cell to 1. Since no read operations are performed, R is equal to –.

The above presented FP formalism leads to a standard representation of a fault model. Test generation and verification processes benefit from this standard representation. Each time that a new fault model is defined, it can be immediately used by those processes thanks to this standard notation. Next sections resort to the presented FP formalism to define the state-of-the-art of functional fault models and the existing test solutions.

1.3.3 Fault Model Classification

According to the number of operations (m) needed in the sequence, the fault is classified as Static ($m \leq 1$) or Dynamic ($m > 1$).

– *Static faults*: For their sensitization, at most one operation (read/write) is required ($m \leq 1$). For example, the simple Stuck-At Fault model (SAF) affecting a memory core-cell is considered. In particular, the faulty cell i is stuck-at zero, meaning that whatever the operation performed on cell i its content remains at logic '0.' In order to sensitize this fault, it is sufficient to set the state of cell i to logic '1.' At the end

of this operation, the cell will not swap its state from '0' to '1' as expected. The FP representation is thus <$i(1)/i(0)/i(-)$>, in this example $m = 0$. Another example is the Transition Fault model, where a memory core-cell fails to undergo a transition ($0 \rightarrow 1$) when it is written. The FP representation is thus <$i(0w_1)/i(0)/i(-)$>, in this example $m = 1$. Other static faults are for example the various types of Coupling Faults (CFs) (Van de Goor 1998).

– *Dynamic faults*: For their sensitization, more than one operation (read/write) in sequence is required ($m > 1$). As shown in the remaining of this book, all parts of the memory may be subject to these faulty behaviors. For example, the Address Decoder Open Faults (ADOFs) are related to dynamic faults in the address decoder (Dilillo et al. 2004a), dynamic Read Destructive Faults (dRDFs) are dynamic faults linked to failures in the core-cell (Dilillo et al. 2004b), Un-Restored Destructive Write Faults (URDWFs) are dynamic faults due to failures either in the pre-charge circuit or in the write driver (Ney et al. 2007).

The class of static faults has been deeply analyzed and several meaningful test solutions have been developed (Van de Goor 1998). Unfortunately, in the latest SRAM technologies, dynamic faults appear as the predominant root cause of errors and hence require new and dedicated test solutions.

Dynamic Faults are modeled by FPs having $m > 1$. The cardinality of the *Dynamic Fault Set* (DFS) is infinite as the number of possible operations is not limited. The DFS is usually split into subsets, each one including the FFMs requiring the same number of operations to be sensitized. Thus, a two-operation DFS identifies the set of dynamic faults that require the application of two memory operations to be sensitized ($m = 2$) whereas the m-operation DFS corresponds to a generic dynamic fault set.

Dynamic Faults can be additionally clustered according to the number of memory cells (| f-cells |) involved in the faulty behavior (as already mentioned in Section 1.3.2). Three main categories are considered in the literature:

– The first one is characterized by having | f-cells | = 1. In this case only one memory cell is involved in the fault. This category is defined as single-cell m-operation DFS.
– The second one is characterized by | f-cells | = 2. In this case two memory cells are involved in the fault. This category is defined as two-cell m-operation DFS. The two involved cells are classified as aggressor cell (a), i.e., the cell that sensitizes a given fault, and victim cell (v), i.e., the cell that exhibits the fault effect.
– The third one occurs when | f-cells | > 2. In this case more than two memory cells are involved in the fault. This category is defined as n-cell m-operation DFS.

In the literature, the single and two-cell two-operation DFS are the only investigated type of dynamic faults (Van de Goor and Al-Ars 2000). The approach followed to identify the target DFS consists in generating all the possible sets of FPs. An exhaustive list of 30 FPs that covers all the single-cell two-operations DFS is given in (Van de Goor and Al-Ars 2000). Only a subset of 12 FPs has been demonstrated

to be realistic (Van de Goor and Al-Ars 2000) leading to a set of three FFMs. The functional fault models considered as realistic are the following:

Dynamic Read Disturb Fault (dRDF): a write operation immediately followed by a read operation changes the logical value stored in the memory cell and returns an incorrect output.

Dynamic Deceptive Read Disturb Fault (dDRDF): a write operation immediately followed by a read operation changes the logical value stored in the memory cell, but returns the expected output.

Dynamic Incorrect Read Disturb Fault (dIRF): a write operation immediately followed by a read operation does not change the logical value stored in the memory cell, but returns an incorrect output.

Table 1.1 shows the FPs that model these FFMs. Left and right columns report the dynamic FFMs and the list of FPs, respectively. Each FFM is composed of four FPs. Since only one memory cell is involved in this type of fault, the notation previously defined in Section 1.3.2 is simplified by omitting the memory cell address. For example, FP $<i(0w_0r_0)/i(1)/i(1)>$ affecting memory cell i is simplified by omitting the information of the memory cell address, i.e., $<0w_0r_0/1/1>$.

Table 1.1 Single-cell two-operation dynamic FFM

FFM	Fault primitives
dRDF	$<0w_0r_0/1/1>$, $<1w_1r_1/0/0>$, $<0w_1r_1/0/0>$, $<1w_0r_0/1/1>$
dDRDF	$<0w_0r_0/1/0>$, $<1w_1r_1/0/1>$, $<0w_1r_1/0/1>$, $<1w_0r_0/1/0>$
dIRF	$<0w_0r_0/0/1>$, $<1w_1r_1/1/0>$, $<0w_1r_1/1/0>$, $<1w_0r_0/0/1>$

The two-cell two-operation DFS is characterized by $|\text{ f-cells }| = 2$ and $m = 2$. In this case, it is necessary to distinguish between the number of operations applied on the aggressor (a) and the number of operations applied on the victim (v). Moreover, it is important to consider the mutual position of aggressor and victim cells, i.e., whether $a < v$ or $v > a$, where '$a < v$' means that the address of the aggressor is lower than the address of the victim. An exhaustive list of 192 FPs that covers all the two-cell two-operations DFS is given in (Van de Goor and Al-Ars 2000). Only a subset of 32 FPs has been demonstrated to be realistic (Van de Goor and Al-Ars 2000) leading to a set of four FFMs. The four functional fault models considered as realistic are the following:

Dynamic Disturb Coupling Fault (dCFds): a write operation immediately followed by a read operation performed on the aggressor causes the victim to flip.

Dynamic Read Disturb Coupling Fault (dCFrd): a write operation immediately followed by a read operation performed on the victim when the aggressor is in a given

state changes the logical value stored in the memory cell, and returns an incorrect output.

Dynamic Deceptive Read Disturb Coupling Fault (dCFdrd): a write operation immediately followed by a read operation performed on the victim when the aggressor is in a given state changes the logical value stored in the memory cell, but returns the expected output.

Dynamic Incorrect Read Disturb Coupling Fault (dCFir): a write operation immediately followed by a read operation performed on the victim when the aggressor is in a given state does not affect the logical value stored in the memory cell, but returns an incorrect output.

Table 1.2 Two-cell two-operation dynamic FFM

FFM	Fault primitives
dCFds	<0w_0r_0,0/1/–>, <0w_0r_0,1/0/–>, <1w_1r_1,1/0/–>, <1w_1r_1,0/1/> <0w_1r_1,0/1/–>, <1w_0r_0,1/0/–>, <0w_1r_1,1/0/–>, <1w_0r_0,0/1/–>
dCFrd	<0,0w_0r_0/1/1>, <1,0w_0r_0/1/1>, <1,1w_1r_1/0/0>, <0,1w_1r_1/0/0> <0,0w_1r_1/0/0>, <1,0w_1r_1/0/0>, <1,1w_0r_0/1/1>, <0,1w_0r_0/1/1>
dCFdrd	<0,0w_0r_0/1/0>, <1,0w_0r_0/1/0>, <1,1w_1r_1/0/1>, <0,1w_1r_1/0/1> <0,0w_1r_1/0/1>, <1,0w_1r_1/0/1>, <1,1w_0r_0/1/0>, <0,1w_0r_0/1/0>
dCFir	<0,0w_0r_0/0/1>, <1,0w_0r_0/0/1>, <1,1w_1r_1/1/0>, <0,1w_1r_1/1/0> <0,0w_1r_1/1/0>, <1,0w_1r_1/1/0>, <1,1w_0r_0/0/1>, <0,1w_0r_0/0/1>

Table 1.2 shows the FPs modeling each FFM. Each FFM is composed of eight FPs. In this case, two memory cells are involved, a and v. The notation is simplified, with respect to the definition of Section 1.3.2, by omitting the involved memory cells. To clearly identify the involved cells, S is divided in subsequences S_A, S_V, where S_A lists the operations applied on the aggressor and S_V lists the operations applied on the victim. FB always refers to the victim cell, which is the cell exhibiting the fault effect. For example, the FP <$a(0w_0r_0)$,$v(0)$/$v(1)$/$v(–)$> affecting memory cells a and v is simplified by omitting the information of the memory cell addresses, i.e., <0w_0r_0,0/1/–>.

The n-cell m-operation DFS characterized by | f-cells | ≥ 2 and $m \geq 2$ has never been considered in the literature. From a theoretical point of view, it is possible to compute the FPs to model this type of dynamic faults, but in practice it has no been

longer considered. Indeed, up to now it has never been demonstrated that this type of DFS is a realistic set of dynamic faults.

The above description has presented the state-of-the-art of dynamic fault models affecting the core-cell formalized using FPs. These fault models have been exploited to obtain meaningful test solutions as discussed in the next section.

Next chapters, which investigate dynamic faults affecting each block of a SRAM, lead to consider an extended set of dynamic faults with respect to the one discussed in this section.

1.3.4 Test Solutions and Algorithms

The first memory tests proposed in the literature were 'ad-hoc' test algorithms. They were developed without resorting to formal fault models (as those described in the previous section) and proofs. Tests such as *Scan*, *Galpat*, and *Walking 1/0* (Van de Goor 1998) belong to this class. The main drawback of these algorithms is the high complexity, generally not linear with respect to the memory size, that makes them not applicable to nowadays large memories (i.e., $O(N^2)$, where N is the size of the memory as defined in Definition 1.1).

To overcome the complexity issues of ad-hoc test solutions, the class of March tests has been introduced and defined in (Van de Goor 1998). The formal definition of a March test is the following:

Definition 1.4 A March test is a test algorithm composed of a sequence of *March Elements*. Each March Element (ME) is a sequence of memory operations applied sequentially on a certain memory cell before proceeding to the next one. The way in which one moves from a certain address to another one is called *Address Order* (AO). The AO characterizes the ME and can be done in either one of two address orders: an increasing ($\Uparrow$) address order (e.g., from address 0 to address $n - 1$) or a decreasing ($\Downarrow$) address order which is the opposite of the $\Uparrow$ address order. When the address order is irrelevant the symbol $\Updownarrow$ is used. Hereinafter, a March test is denoted by using a '{...}' bracket and a March Element using a '(...)' bracket. The ith operation is defined as op_i where $op_i \in \{wd, rd\}$, $d \in \{0,1\}$ in which '*wd*' means 'write logic value d in the memory cell' and '*rd*' means 'read the content of the memory cell and verify that its value is equal to *d*.'

In general, the complexity of a March test is in the order of $O(N)$, i.e., linear with respect to the size of the memory under test. More precisely, it is defined as the number of memory operations composing the different March elements multiplied by the size of the memory (N). Figure 1.5 shows an example of a March test, namely the MATS+ (Van de Goor 1998), which is composed of three March elements (M0,

$$\{\ \Updownarrow(w0)\ \ \Uparrow(r0, w1)\ \ \Downarrow(r1, w0)\}$$
$$\text{M0} \qquad \text{M1} \qquad \text{M2}$$

Fig. 1.5 March test MATS+

M1, and M3). The complexity of March MATS+ is 5 N as it contains five operations. The MATS+ is able to detect only static faults, in particular the Stuck-At faults and the Address faults (Van de Goor 1998).

March tests are nowadays the dominant type of memory tests used in the industry due their low complexity. Despite the high fault coverage achieved by March tests with respect to the set of static fault (Van de Goor 1998), non-classical faults that are based on a more physical view of the memory are not covered by March tests. For example, the Un-restored Write Fault affects the pre-charge logic of a memory cell column (Niggemeyer et al. 1998). This type of fault requires a particular set of operations to be detected. This set of operations leads to violate the definition of a March test. In order to increase the coverage capabilities of a March test, a deeper analysis of its definition leads to a set of so-called Degrees Of Freedom (DOF) presented in Niggemeyer et al. (1998) and summarized below:

1. The address sequence of a March test can be freely chosen as long as all addresses occur exactly once and the sequence is reversible.
2. The address sequence for the initialization of the memory can be freely chosen as long as all addresses occur at least once.
3. If the March test is built symmetrically, for example to detect both stuck-at-0 and stuck-at-1 faults, the data written to the cells can be inversed.
4. The data within a read/write operation do not necessarily have to be equivalent for all memory addresses as long as the detection probabilities for basic faults are not affected.
5. The input data is not defined during read operations.
6. The output data is not defined during write operations.

The above degrees of freedom have been exploited in Niggemeyer et al. (1998) to obtain a meaningful March test targeting some non-classical static faults. In the next chapters, the same idea will be used to extend the fault coverage of some March tests and hence to cover new dynamic faults. The main advantage given by the usage of DOFs is that a March test may have a higher fault coverage without increasing its complexity (i.e., without adding operations).

In the literature, few March tests target the whole set of dynamic faults showed in Tables 1.1 and 1.2. Table 1.3 reports the state-of-the-art of March tests that cover dynamic fault models presented in this chapter. The first column lists the name of the March test. The second column details the March algorithm. The third column gives the complexity of each algorithm, and the last column reports the set of faults covered by the corresponding algorithm.

In Hamdioui et al. (2002), authors present two March test algorithms called RAW1 and RAW, respectively (see Table 1.3). The former one has a complexity of 13N and covers the class of realistic single-cell dynamic faults. The latter has a complexity of 26N and covers both classes of realistic single-cell and realistic two-cell two-operation dynamic faults. In Benso et al. (2005), two March test algorithms are proposed (Table 1.3). The March AB1 with a complexity of 11N covers the class of realistic single-cell dynamic faults. The March AB with a complexity of

Table 1.3 March tests targeting dynamic fault models

Test name	March test	$O(N)$	FFM
RAW1	{⇕($w0$) ⇕($w0$,$r0$) ⇕($r0$) ⇕($w1$,$r1$) ⇕($r1$) ⇕($w1$,$r1$) ⇕(r1) ⇕($w0$,$r0$) ⇕($r0$)}	13 N	Single-cell two-operation Dynamic Faults
RAW	{⇕($w0$) ⇑($r0$,$w0$, $r0$,$r0$,$w1$,$r1$) ⇑($r1$,$w1$,$r1$,$r1$, $w0$,$r0$) ⇓($r0$,$w0$,$r0$,$r0$, $w1$,$r1$) ⇓($r1$,$w1$,$r1$,$r1$, $w0$,$r0$) ⇕($r0$)}	26 N	Single and two-cell two-operation Dynamic Faults
AB1	{⇕($w0$) ⇕($w1$,$r1$,$w1$,$r1$,$r1$) ⇕($w0$,$r0$,$w0$,$r0$,$r0$)}	11 N	Single-cell two-operation Dynamic Faults
AB	{⇕($w1$) ⇓($r1$,$w0$,$r0$,$w0$,$r0$) ⇓($r0$,$w1$,$r1$,$w1$,$r1$) ⇑($r1$,$w0$,$r0$,$w0$,$r0$) ⇑($r0$,$w1$,$r1$,$w1$,$r1$) ⇕($r1$)}	22 N	Single and two-cell two-operation Dynamic Faults
MD2	{⇕($w0$) ⇑($r0$,$w1$,$w1$,$r1$,$w1$,$w1$, $r1$,$w0$,$w0$,$r0$,$w0$,$w0$, $r0$,$w0$,$w1$,$w0$,$w1$) ⇓($r1$,$w0$,$w0$,$r0$,$w0$,$w0$, $r0$,$w1$,$w1$,$r1$,$w1$,$w1$, $r1$,$w1$,$w0$,$w1$,$w0$) ⇑($r0$,$w1$,$r1$,$w1$,$r1$, $r1$,$r1$,$w0$,$r0$,$w0$,$r0$, $r0$,$r0$,$w0$,$w1$,$w0$,$w1$) ⇓($r1$,$w0$,$r0$,$w0$,$r0$, $r0$,$r0$,$w1$,$r1$,$w1$, $r1$,$r1$,$r1$,$w1$,$w0$, $w1$,$w0$) ⇕($r1$)}	70 N	All the possible two-operation Dynamic Faults

22N covers both classes of realistic single-cell and realistic two-cell two-operation dynamic faults. Finally, in Harutunyan et al. (2007) authors proposed a March test algorithm MD2 with a complexity of 70N, targeting the whole set of possible FPs describing both single and two-cell two-operation dynamic faults.

The above presented test solutions have been proved to cover dynamic faults presented in Section 1.3.3. However, a common feature of the test solutions proposed so far is that they always embed all the operations required for the sensitization of dynamic faults. For example, if a dynamic fault is sensitized by five consecutive read operations, the resulting March test will contain at least one March element containing those five read operations. The direct consequence is that as the number of operations required to sensitize a fault increases, the complexity of the March test proportionally increases. Moreover, the presented March tests always aim at covering the largest set of faults as possible. The consequence is that these approaches may consider a set of faults even containing those faults with a very low probability to appear. Again, this situation leads to obtain very effective test solutions but with a higher complexity, thus leading to a higher test cost.

In order to minimize the test development effort and produce more efficient and dedicated memory tests, it is possible to rely on information provided by preliminary electrical analysis. In this context, the analysis of the memory layout allows to determine the realistic defect sites within the memory elements. Then, electrical simulations of defects allow determining those that may lead to a faulty behavior of the SRAM. The next step is the modeling of the faulty behavior of the memory and the generation of effective test algorithms. Next chapters show how this bottom-up methodology (from the electrical to the logical level) works for each memory block.

1.4 Test Generation

Several March tests targeting different sets of memory faults (static and dynamic) have been proposed in the literature. Most of them have been generated by hand. Due to the occurrence of new and more complex dynamic fault models, the task of writing March tests manually is becoming a harder and time consuming task. Moreover, March tests generated by hand may not lead to optimal results (i.e., maximum fault coverage with the lowest complexity) or they can lead to optimal results in a higher time compared to the automatic generation. For example, March RAW presented in Section 1.3.4, with a complexity of 26N, has been manually generated whereas March AB generated automatically reduces the complexity of the March test by 18% with respect to March RAW maintaining the same fault coverage.

Figure 1.6 shows the structure of an Automatic March test Generation (AMTG) tool. The AMTG is generally based on a Memory Model to describe the behavior of the cell array, and a target Fault List that contains a set of fault models to describe the faulty behaviors that the algorithm is designed to detect. Examples of Memory Models used by the AMTG can be found in Section 1.3.1. The fault list can be described resorting to the fault primitives presented in Section 1.3.2. The quality of

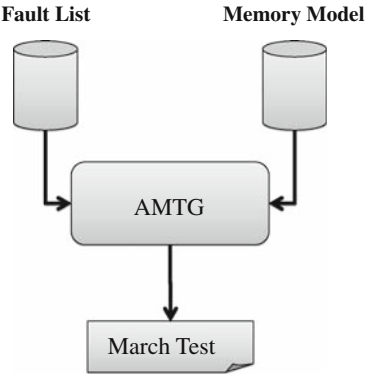


Fig. 1.6 Automatic March test generation

a test algorithm is then evaluated based on its fault coverage and its length, which directly influences the final test time.

In Van de Goor and Smit (1994), authors present a methodology for generating non-redundant March tests. The proposed algorithm uses state machines to mathematically model both good and faulty memories. The March test generation process is based on the use of a transition tree. The tree is built in such a way that different paths (sequences) from the root to the leaves represent lists of operations needed to sensitize and detect a given list of faults. Each sequence can thus be translated into different March tests. The size of the tree, theoretically unbounded but in fact limited by a user-defined upper bound, depends on how many operations are necessary to cover all the faults in the list. When the tree creation ends, the shortest sequence that covers all the targeted faults is selected and translated into a March test.

The main drawback of this approach is the need to define an upper bound of the tree size. If the upper bound is too low the algorithm can fail in finding a solution. If it is too high, the computation time becomes unacceptable. Finding the appropriate upper bound usually means making several iterations, thus decreasing efficiency. In addition, this method performs an exhaustive search to find the shortest path on the transition tree. As the size of the transition tree increases, the algorithm becomes more and more inefficient.

In Zarrineh et al. (1998), authors resort to the same model presented in Van de Goor and Smit (1994) to represent both good and faulty memories, but instead of resorting to transition trees, each fault in the fault list is covered by a so-called Primitive March test (PMT). Different PMTs are then combined using a set of combination rules to generate all possible March tests able to cover the given fault list. This methodology does not need to perform a search procedure on a tree, and therefore is more efficient in terms of computational time than the one proposed in Van de Goor and Smit (1994).

In Al-Harbi and Gupta (2001), the generation methodology is still based on a transition tree but a branch-and-bound framework is used, in which March Elements

are added to a partial test to limit the search space. A fault-collapsing procedure is then performed to reduce the fault list complexity. The starting node of the tree is an empty March test, which does not contain any March element. The algorithm starts creating new nodes, and adding operations needed to detect faults. Each new node is checked to evaluate the actual coverage, and nodes with the best properties in terms of coverage and number of operations are selected to continue the generation process. The search process stops when all faults are covered.

Cheng et al. (2002) presents a completely different approach, named Test Algorithm Generation by Simulation (TAGS). It is based on fault simulation and can deal with both bit-oriented and word-oriented memories. The algorithm starts from an empty March test, and during each iteration it adds new elements. An element can be either a March Element or a Memory Operation. Every time that a new element is added, then the resulting March test is simulated by using the memory fault simulator tool developed by the same authors and described in Wu et al. (1999). After the simulation, the achieved fault coverage is compared to the previous one. If the coverage is not improved, the added element is dropped. This process is repeated until the test reaches the required fault coverage or a test length upper bound.

The algorithm presented in Niggemeyer and Rudnick (2004) resorts to the same concept of transition tree already proposed in Van de Goor and Smit (1994). Conversely to Van de Goor and Smit (1994), a complete path from the root to a leaf represents a single March Element and not a complete March test. In this approach, each March Element is simulated in order to determine its fault coverage. Those having a low fault coverage are dropped while the remaining are assembled, using a set-covering algorithm, to compose the final March test.

Benso et al. (2008) presents a March test generation algorithm able to manage both static and dynamic faults. This approach resorts to a memory model based on a graph representation. The vertexes of the graph represent the memory cell states while the edges represent the fault models. March tests are generated by traversing each edge of the graph.

This section presented the state-of-the-art of the automatic March test generation algorithm. The presented algorithms address both static and dynamic faults. Moreover, automatic test generation leads to less complex March tests with respect to manually generated tests, thus allowing the reduction of the overall test time. The next section describes how the fault coverage of a given March test can be determined in an automatic way.

1.5 Test Validation

The main issue in memory testing is to define comprehensive fault models able to carefully represent the most common defects occurring in the production phase of the chips. Along with fault models, test algorithms have to be developed and validated. Manual validation of March tests become less and less feasible due to the

increasing complexity of both March tests and fault models. Memory fault simulation is therefore necessary to compute the Fault Coverage of a test sequence every time a new defect is discovered and the corresponding fault model is defined. Moreover, memory fault simulation is also used to build the fault dictionary required to perform memory diagnosis.

Figure 1.7 gives the architecture of a typical memory fault simulator. It requires three main inputs: (i) the Fault List, (ii) the applied March test, and (iii) the Memory Model. The results of the fault simulation are the Coverage Report and the Fault Dictionary.

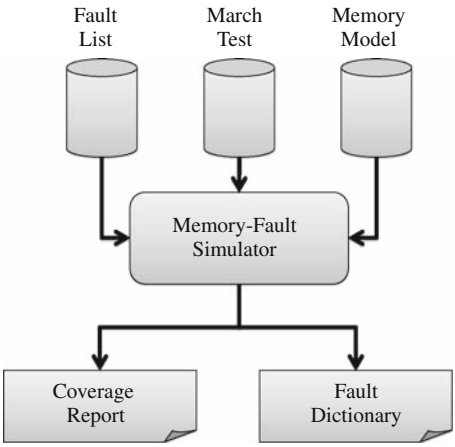


Fig. 1.7 Memory fault simulator architecture

The memory fault simulation is based on the serial fault simulation paradigm. The fault **F** is injected into the memory and the March test is applied. The values of the read operations of the simulated March tests are used to monitor the memory behavior in the presence of **F**. By this way, the fault coverage achieved by the simulated March test is computed for the injected fault **F**. This procedure is applied for the entire fault list.

The complexity **C** of a fault simulation algorithm can be calculated as follows:

$$\mathrm{C} = \mathrm{F} \times \mathrm{MTL}, \tag{1.3}$$

where **F** is the number of faults in the fault list and **MTL** is the length of the March test. The fault number is strictly related to the size of the considered memory. For example, considering a memory with a size equal to N cells means that there are N possible single cell faults and N^2 possible two-cell faults. The same consideration can be done for the march test length that is equal to O(N). The conclusion is that the complexity of the memory fault simulation is equal to O(N^3).

This analysis seems to limit the feasibility of the memory fault simulation to a very small memory. But thanks to the regular structure of the memories, it is not necessary to simulate a big memory to compute the fault coverage of a given

March test. Indeed, the fault coverage of a March test will be the same for a 1 KIbytes memory than for a 16 Mbytes memory. Moreover, if only single- and two-cell faults are considered during fault simulation, only a two-bit memory is sufficient to compute the fault coverage. Thus, the fault-simulation time is drastically reduced.

Based on this principle, the main tools proposed in the literature to perform memory fault simulation are those presented in Riedel and Rajski (1995), Wu et al. (1999), and Benso et al. (2002).

1.6 Conclusion

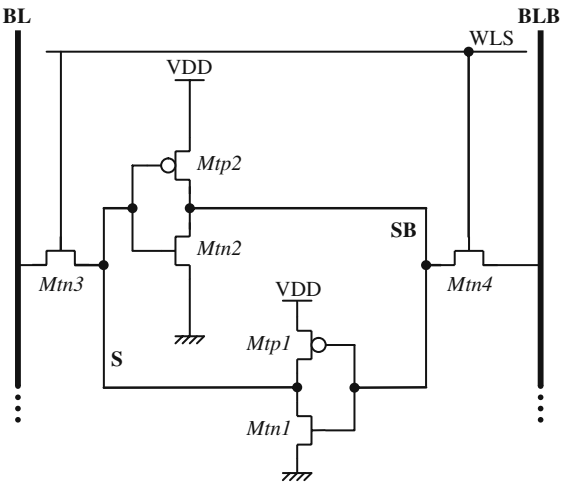
The aim of this chapter has been to first present an overview about the semiconductor memories and then to introduce the main issues related to SRAM testing. In the first part of the chapter, the basic structure of an SRAM has been introduced. In the second part, the basic notions of memory testing have been presented. In particular, fault models have been described using the fault primitive formalism. Based on this formalism, the taxonomy of dynamic faults as well as the state-of-the-art of existing test solutions has been discussed. Finally, automatic test generation and test validation have been discussed in the last part of this chapter.

Chapter 2
Resistive-Open Defects in Core-Cells

2.1 The SRAM Core-Cell

Figure 2.1 depicts the typical six-transistor SRAM core-cell in CMOS technology. The circuit consists of a flip-flop based on two cross-coupled inverters and two access transistors Mtn3 and Mtn4. The access transistors are turned ON when the word line signal is selected (WLS at VDD), and they connect the flip-flop to the bit lines BL and BLB. Note that both bit lines BL and BLB are used for read/write purpose. The access transistors act as transmission gates allowing bi-directional current flow between the flip-flop and the bit lines.

Fig. 2.1 Scheme of core-cell



2.1.1 Reading in the Core-Cell

In order to detail the read operation, let us consider a core-cell storing a logic '1' with node S at VDD and node SB at 0 V. Before the beginning of the read operation, the bit lines BL and BLB are pre-charged to a certain fixed voltage, commonly at

A. Bosio et al., *Advanced Test Methods for SRAMs*,
DOI 10.1007/978-1-4419-0938-1_2, © Springer Science+Business Media, LLC 2010

VDD. When the word line is selected (WLS at VDD), the two access transistors Mtn3 and Mtn4 are turned ON, node S and bit line BL are at the same potential (VDD), and no current flow occurs. On the other side of the circuit, the current flows from bit line BLB (charged at VDD) through transistors Mtn4 and Mtn2, provoking the discharge of bit line BLB. The relevant parts of the circuit during a read operation are shown in Fig. 2.2, where the capacitors C_{BL} and C_{BLB} are the equivalent capacitances of bit lines BL and BLB, respectively.

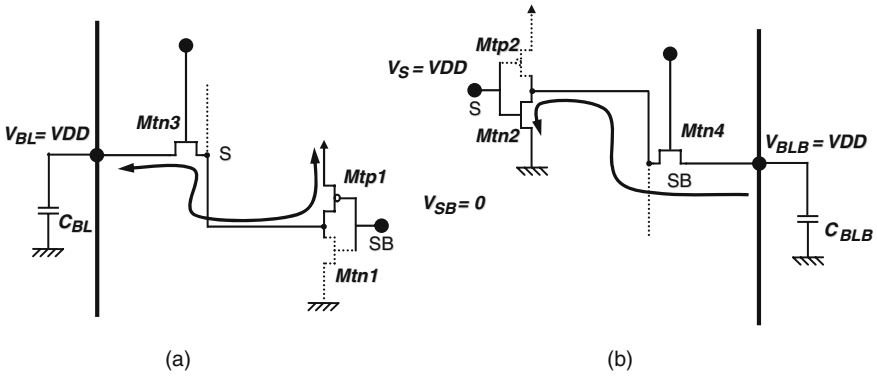


Fig. 2.2 Relevant parts of SRAM core-cell circuit during a read operation ($r1$)

During a read '1' ($r1$) operation, the voltage across C_{BLB} decreases while C_{BL} remains quite stable at VDD. Thus, a differential voltage ΔBL occurs between bit lines BL and BLB. Usually, a few hundreds of millivolt differential voltage is sufficient for the sense amplifier to operate a correct read operation. Furthermore, the core-cell must be designed to remain stable and keep its logic state during readout, for the voltage excursions on nodes S and SB. Typically, the inverters are designed in a way that PMOS and NMOS transistors match with the inverter threshold at VDD/2. The access transistors are usually made two or three times wider than the NMOS transistors of the inverters.

2.1.2 Writing in the Core-Cell

In order to detail the write operation, let us consider a core-cell storing a logic '0' (node S at 0 V and node SB at VDD) in which a write '1' ($w1$) is operated. For the $w1$ operation, the write driver sets bit line BL at VDD and bit line BLB at 0 V. The core-cell is selected with word line signal WLS at VDD. Figure 2.3 shows the relevant parts of the circuit during the $w1$ operation: node S is pulled up toward the threshold voltage VDD/2 (see Fig. 2.3a) and node SB is pulled down toward VDD/2 (see Fig. 2.3b).

In Fig. 2.3, the capacitors C_S and C_{SB} are the parasitic capacitances at nodes S and SB, respectively. The regenerative feedback, which causes the flip-flop to

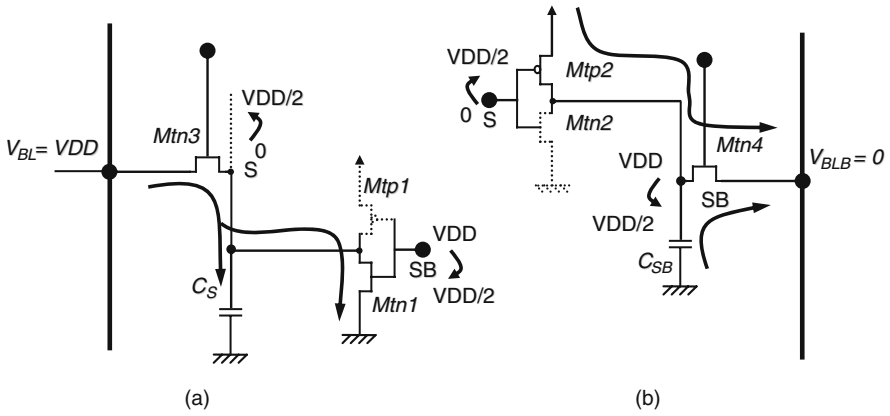


Fig. 2.3 Relevant part of the SRAM core-cell circuit during a write operation ($w1$)

switch, will begin when either V_S or V_{SB} reaches VDD/2. Once this condition is satisfied, the positive feedback takes over.

2.2 Analysis of Resistive-Open Defects in the Core-Cell

The study on the influence of resistive-open defects in the SRAM core-cell is generally made by injecting defects in the circuit itself. For simplicity, only one defect can be considered for each analysis, because the occurrence of multiple defects has a low probability in such small circuit composed of only six transistors.

2.2.1 Defect Location

The defect injection in the SRAM core-cell is depicted in Fig. 2.4 . The resistive-open defects are placed on the interconnections of the circuit. Due to the symmetry of the structure, these six locations allow an exhaustive analysis of the resistive-open defects within the core-cell structure (Borri et al. 2005).

2.2.2 Defect Incidence Analysis

Each injected defect induces a faulty behavior during the memory operation (read/write) as well as data retention problems in hold mode (Dilillo et al. 2004). The following analysis refers to the core-cell scheme in Fig. 2.4.

– *Defect Df1*: This defect produces a delay in the charging/discharging operation of node SB during the write operations. The defect essentially leads to a transition fault (TF). This fault is static and many common March tests are able to

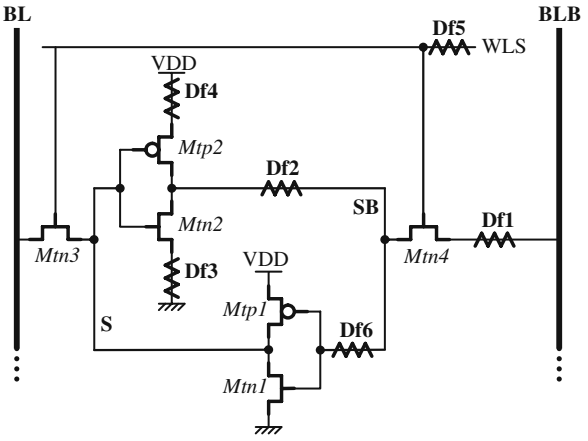


Fig. 2.4 Resistive-open defects injected into the memory core-cell

detect it. Defect Df1 is on the path for the core access also for the read operation and may lead to (dynamic) read destructive faults when the core-cell stores a logic '0.'

– *Defect Df2*: This defect induces a delay in the output of INV1 especially during the discharge of node SB. This delay may cause a Read Destructive Fault (RDF). During the $r1$ operation (node S at '1' and node SB at '0'), BLB is pre-charged at VDD, and for a certain time it pulls up bit line SB that is at logic '0.' The equivalent capacitance of bit line BLB is much larger than the equivalent capacitance of core-cell node at SB, and the pull-up of BLB is not well counterbalanced by the pull-down of INV1 because of the resistive-open defect. For this reason, the read operation may cause the switching of INV2 and so the swap of the core-cell content. In some cases, the data loss does not involve an incorrect read; thus a further read operation is necessary to observe the fault (Deceptive Read Destructive Fault, DRDF).
– *Defect Df3*: This defect produces effects similar to those of defect Df2. For the faults RDF and DRDF, induced by defects Df2 and Df3, the sensitization pattern is $0w1r1$ that does not need to be applied at-speed (Borri et al. 2005). This constraint is useful for the choice of the fault detection strategy. Defect Df3 is in the refreshment loop and may lead to a dynamic Read Destructive Fault (dRDF).
– *Defect Df4*: This defect is placed in the pull-up of INV1 and may induce static and dynamic faults (Zarrineh et al. 2000, Dilillo et al. 2005a). For large resistance values of Df4, static RDFs and DRDFs occur as for defect Df2. Very large values of resistance lead to spontaneous data loss of the core-cell: Data Retention Faults (DRFs). For low values of resistance, multiple consecutive read operations may lead to core-cell content swap (dynamic RDFs). In the same range of resistance, defect Df4 may also cause dynamic DRFs (dDRF).
– *Defect Df5*: This defect may represent the resistive effect of long connection wires as word lines. Defect Df5 implies incorrect read faults (IRFs) and TFs. IRFs and

TFs are static faults. These two faults occur because read and write operations need a certain minimal time to be performed. During these operations, nodes S and SB are connected to the bit lines BL and BLB by the pass-transistors Mtn3 and Mtn4. The defect involves a delay in the switching ON of these two transistors, reducing the operative time of the read/write operations. The read operation needs a time longer than a write operation to be acted; thus IRFs appear for smaller resistance values than the TFs.

– *Defect Df6*: This fault is at the input of INV2 and involves TFs. It appears for high values of resistance because the defect is placed at the gates of the two transistors of INV2. No bias current enters into the MOS transistor gate; thus the resistive defect has to be very large to generate large delay. The TF occurs during the write operations. Defect Df6 causes a delay for both the operations of pull-up ($w1$, '1' on BL and '0' on BLB) and pull-down ($w0$, '0' on BL and '1' on BLB) of INV2; thus the write operation may fail. For $w0$ and $w1$ operations, there can be a different resistance threshold for the faulty behavior, because during a write operation in an SRAM core-cell, the first to switch is the node where a logic '0' is forced (Sedra and Smith 1998). Consequently, the incorrect writing is more likely to occur when a logic '0' is forced on BLB, i.e., during a $w1$ operation. Defect Df6 is in the refreshment loop and may lead to a dynamic RDF.

Table 2.1 presents a summary of the fault models identified for each injected resistive defect.

Table 2.1 Fault models identified for each injected defect

Defect	Fault model
Df1	TF, dRDF
Df2	DRDF, RDF, dDRF, dRDF, dDRDF
Df3	DRDF, RDF, dDRF, dRDF, dDRDF
Df4	DRDF, RDF, dRDF, dDRDF
Df5	IRF, dRDF, TF
Df6	TF, dRDF

Definitions of fault models reported in Table 2.1 are the following:

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.

Read Destructive Fault (RDF) (Adams and Cooley 1996): A core-cell is said to have an RDF if a read operation performed on the core-cell changes the data in the core-cell and returns an incorrect value on the output.

dynamic Read Destructive Fault (dRDF) (Van de Goor and Al-Ars 2000, Hamdioui et al. 2002): 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 an incorrect value on the output.

Deceptive Read Destructive Fault (DRDF) (Adams and Cooley 1996): A core-cell is said to have a DRDF if a read operation performed on the core-cell returns the correct logic value, but it changes the contents of the core-cell.

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

Data Retention Fault (DRF): A DRF occurs when a memory core-cell loses its previously stored logic value after a certain period of time during which it has not been accessed (Van de Goor 1998).

dynamic Data Retention Fault (dDRF): A dDRF occurs when a memory core-cell loses its previously stored logic value after at least two read or write operations performed on other core-cells.

The fault models affecting the core-cell can be classified into two groups. The first group includes static faults while the second one includes dynamic faults. The dynamic faults appear more frequently for defects placed in the loop of two inverters composing the core-cell. These defects disturb the self-refreshment of the stored value.

2.2.3 Simulation Set-Up and Results

Electrical simulations of the SRAM core-cell affected by resistive-open defects have been presented in Borri et al. (2005), Dilillo et al. (2005a) and Dilillo et al. (2005b) and reported here. These simulations have been performed on 8K $\times$ 32 memories, organized in an array of 512 word lines $\times$ 512 bit lines, in a 130 nm technology. In order to reduce the simulation time, the considered memories are simplified versions that include a reduced set of core-cells and all the peripheral circuits as pre-charge devices, sense amplifiers, write drivers, output buffer, and the column and row address decoders. The operating environment is set with the following parameters:

- *Process corner:* slow, typical, fast
- *Supply voltage:* 1.35 V, 1.5 V, 1.6 V
- *Temperature:* –40°C, 27°C, 125°C

The resistance values of the defects are in a range from few ohms up to several Mohms, since a large range of values have been reported to be realistic within the core-cell (Rodriguez et al. 2002).

Table 2.2 shows a summary of the fault models identified for each injected resistive-open defect according to the conditions that maximize the fault detection, i.e., the minimum detectable resistance value. In Table 2.2, the first column (Dfi) indicates the defect location in the core-cell (*c.f.* Fig. 2.4). The following four columns correspond to the values of the parameters that maximize the fault detection. The last column gives the fault models corresponding to each defect.

Table 2.2 Summary of worst-case PVT corners for the defects of Fig. 2.4 and corresponding minimum detected resistance and fault models

Defect	Process corner	Voltage (V)	Temp (°C)	Min res (Ω)	Observed fault models
Df1	Fast	1.6	–40	∼25k	TF
Df2	Fast	1.6	–	∼8k	RDF/DRDF/(dDRF)
Df3	Fast	1.6	125	∼3k	RDF/DRDF/(dDRF)
Df4	Fast	1.6	125	∼130k/∼130k/80 M	dRDF/dDRF/DRF
Df5	Fast	1.6	–40	100k/140 k	IRF/TF
Df6	Fast	1.6	125	∼2 M	TF

The experiments demonstrate that the detection sequences $1w0r0$ and $0w1r1$ are the most effective for sensitization of the faults linked to the core-cell resistive-open defects (Borri et al. 2005). For defects Df2 and Df3 in Table 2.2, dDRFs are filled in parentheses because, as detailed in Section 2.5, these faults appear in particular conditions. Although defect Df6 is in the loop, it presents a static behavior.

2.3 Analysis and Test of dRDF

The simulation results presented in Table 2.2 show that dynamic Read Destructive Faults (dRDFs) occur in the presence of defect Df4. According to its definition (Van de Goor and Al-Ars 2000), this fault is detectable by a write operation on the affected core-cell followed by a read operation on the same core-cell. However, in case of low resistive defects, multiple read operations in sequence are needed to sensitize the fault (Dilillo et al. 2004b, Borri et al. 2005). This statement is confirmed by the waveforms presented in Fig. 2.5 which show that, after a $w0$ operation, a sequence of five $r0$ operations are needed to sensitize a fault due to defect Df4 = 1.5 MΩ.

The results of the incidence of dRDF with respect to the cycle time and the resistive value of defect Df4 are summarized in the graph of Fig. 2.6 where each point corresponds to a determined couple of values (clock cycle period, resistance value of defect Df4) and is placed in a certain area corresponding to a sensitization sequence like $1w0r0^M$, where $M = 1$–5.

The graph in Fig. 2.6 can be read horizontally (by fixing a certain resistance value) and vertically (by fixing a certain clock cycle period). In the first case, for a certain resistance value, the reading shifts horizontally and it can be observed that, for the same resistance value, large clock cycle periods correspond to a large number of read operations needed for the sensitization. In other words, for lower frequency, the incidence of dRDFs is lower. On the other hand, for a fixed clock cycle period, large resistance values of defect Df4 need a small number of read operations for dRDF sensitization. Moreover, we can observe that the fault sensitization is two times more effective when passing from $1w0r0$ to $1w0r0^5$.

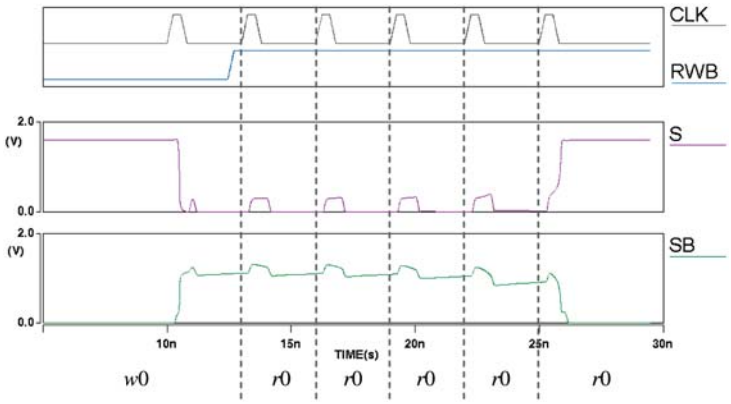


Fig. 2.5 A destructive read occurring after the fifth consecutive *r*0 operation (process corner typical, $T = 125°C$, $V = 1.6$ V, Tcyc $= 3$ ns, $R = 1.5$ MΩ)

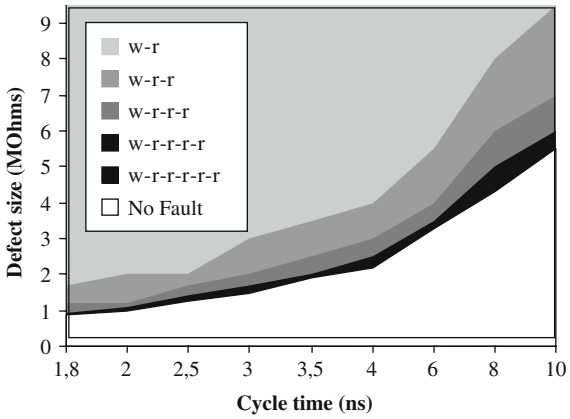


Fig. 2.6 Fault detection as a function of the cycle time and defect size (process corner typical, $T = 125°C$, $V =$ 1.6 V)

2.3.1 Functional Fault Modeling of dRDF

In this section, the fault model of the dRDF has been introduced and detailed with experimental results. The functional fault model of dRDF is composed of four Fault Primitives (FPs) divided into two groups depending on the defect Df4 and its symmetric on inverter INV2.

The first group corresponds to defect Df4 and uses essentially *w*0 and *r*0 operations:

– *FP1*: $<0w_0r_0{}^M/1/1>$ A logic '0' is initially stored in the core-cell. A *w*0 operation immediately followed by *M* *r*0 operations causes the swap of the core-cell content, and a logic '1' is observed at the memory output. *M* is the number of read operations performed after the write operation ($M \geq 1$).

– *FP2*: $<1w_0r_0{}^M/1/1>$ A logic '1' is initially stored in the cell. A $w0$ operation immediately followed by M $r0$ operations causes the swap of the core-cell, and a logic '1' is observed at the memory output. M is the number of read operations performed after the write operation ($M \geq 1$).

The second group of FPs corresponds to the symmetric defect and uses essentially $w1$ and $r1$ operations:

– *FP3*: $<1w_1r_1{}^M/0/0>$ A logic '1' is initially stored in the cell. A $w1$ operation immediately followed by M $r1$ operations causes the swap of the core-cell content, and a logic '0' is observed at the memory output. M is the number of read operations performed after the write operation ($M \geq 1$).
– *FP4*: $<0w_1r_1{}^M/0/0>$ A logic '0' is initially stored in the cell. A $w1$ operation immediately followed by M $r1$ operations causes the swap of the core-cell content, and a logic '0' is observed at the memory output. M is the number of read operations performed after the write operation ($M \geq 1$).

2.3.2 RES: Read Equivalent Stress

The dRDF can be the consequence of resistive-open defects in the core-cell of SRAMs. In particular, it has been empathized above that, in presence of resistive-open defect Df4 depicted in Fig. 2.4, the action of single or multiple read operations immediately after a write operation may cause the swap of the core-cell content. In Dilillo et al. (2004b) and Dilillo et al. (2005b), the authors show that a core-cell can undergo a stress equivalent to a read operation (Read Equivalent Stress, RES) when a read/write operation is performed on other core-cells of the same row (connected to the same word line) and that RESs are more effective to sensitize dRDFs than actual read operations.

When a core-cell is selected for a read/write operation, the pre-charge circuit is turned off in its bit line. For the bit lines that are not involved in the operation, the pre-charge circuit is commonly left ON (Adams 2002). The word line signal is high not only for the selected core-cell, but also for all the other core-cells of the same row. These indirectly selected core-cells are connected with pre-charge circuits that set the bit lines at VDD. Consequently, a read/write operation performed on a core-cell of a certain row causes a pre-charge-induced stress on all the other core-cells of the same row, and this stress is equivalent to the stress produced by an actual read operation on a core-cell. In fact, during a read action, the perturbation of the core-cell is produced by the state of the two bit lines which have been previously set and left floating at VDD, while in the other case (RES) the core-cell is stressed by the same bit lines set at VDD but with the pre-charge circuit still ON.

For clarity, an example is introduced. Figure 2.7 depicts a section of an SRAM block, and in particular the first six core-cells of the ith row. Let us assume that the

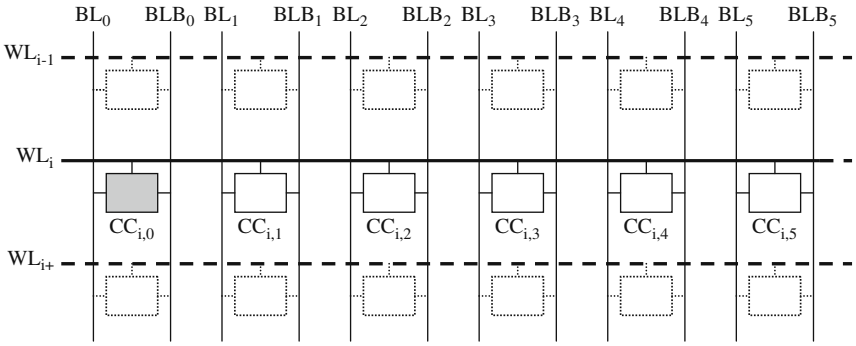


Fig. 2.7 A portion of an SRAM block

first core-cell of the *i*th row ($CC_{i,0}$) is affected by a resistive-open defect in the pull-up transistor of one of the two inverters (Df4 in Fig. 2.4). This defect may cause a dRDF that is normally detectable with the sequence of operations $1w0r0^M$ on core-cell $CC_{i,0}$. Applying the RES principle, a similar sensitization can be obtained by the following sequence: write operation in core-cell $CC_{i,0}$ followed by a sequence of read/write operations acted on the other core-cells of the same row, $1w0\text{RES}^M$. This is possible because, if for example core-cell $CC_{i,1}$ is selected, the pass-transistors (Mnt3 and Mnt4 in Fig. 2.1) of all the core-cells on the same row, in particular the faulty core-cell $CC_{i,0}$, are ON (saturated). So, $CC_{i,0}$ fights against the pre-charge circuit that is in the state ON. Consequently, the faulty core-cell $CC_{i,0}$ undergoes RES stresses similar to actual read operations. Figure 2.8 depicts an example of a

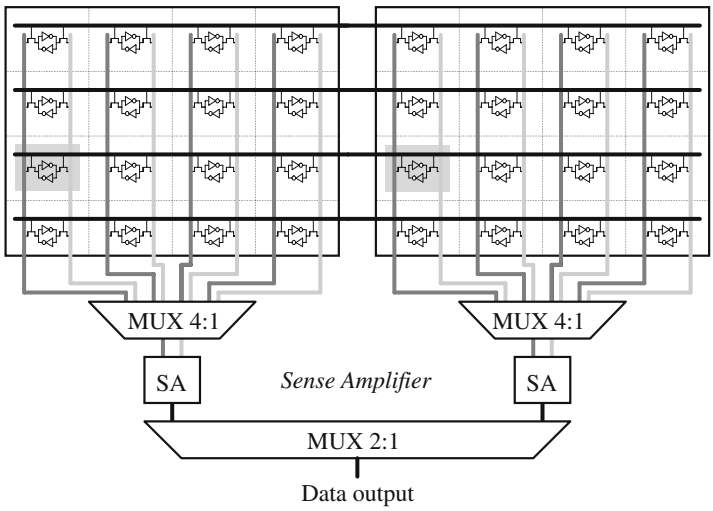


Fig. 2.8 Scheme of a two-block SRAM

two-block SRAM. In reality, when a read operation is performed on a certain core-cell, this operation is actually performed on all the corresponding core-cell of each block, and the actual output is chosen with a further selection made by multiplexers.

Dilillo et al. (2004b) and Dilillo et al. (2005b) give a formal confirmation of the previous assumptions and analysis through SPICE simulations. For these simulations, a reference 8kx32 memory in a 130 nm technology is considered. This memory is organized as an array of 512 word lines $\times$ 512 bit lines, with 128 blocks. When a word line is activated, all the 512 core-cells of the corresponding row are connected to respective bit lines, but only one core-cell per block (128) is accessed for read/write operation and the others (384 = 512–128) undergo a RES.

The simulations estimate and compare the stresses produced in the following situations:

a. On the faulty core-cell, a $w0$ operation is performed immediately followed by $r0$ operations.
b. On the faulty core-cell, a $w0$ operation is done immediately followed by read (b1) or write (b2) operations on the core-cells placed on the same word line.
c. On the faulty core-cell, a $w0$ operation is performed immediately followed by read or write operations on the core-cells on the same word line, but placed in other blocks in the same position than that of the faulty core-cell (highlighted core-cells in Fig. 2.8).

The waveforms in Fig. 2.9 refer to the electrical simulations of a core-cell affected by defect Df4 of value 1.4 MΩ.

The waveforms in Fig. 2.9a represent the control signals; the clock (CLK), the read/write selection (RWB), and the word line and bit line enable signals (WLEN0, WLEN1, and BLEN0). The voltage values of nodes S and SB (see core-cell in Fig. 2.1) are represented in Fig. 2.9b for cases a and b (b1 and b2) and in Fig. 2.9c for cases a, b1, and c. These waveforms show that after a $w0$ operation the fault-free inverter of the core-cell has its output (node S) normally switched to logic ‘0,’ which is an actual electrical power of 0 V. The other inverter has its output switched to logic ‘1,’ which does not correspond to an exact VDD value due to the presence of defect Df4 in its pull-up.

In all cases a, b, and c, the stress action (actual read operations or RESs) performed after the $w0$ operation results in an abnormal swap of the defective core-cell content after two cycles. This confirms the assumption made above, i.e., the effects produced by the RES in terms of sensitization of dRDF are equivalent to actual read stresses on defective core-cells. The waveforms in Fig. 2.9b, c show that in all cases the RES action (b1, b2, and c) produces effects very similar to actual read operations. It can also be observed that in case of read operation the word line enabled signal is ON for a time a little bit longer than for the write operation. This involves that in case b1 the stress is more extended in time and the core-cell flips before.

In order to evaluate the RES in terms of sensitization performance, Dilillo et al. (2005b) proposes parametrical simulations varying the clock period and the resistive

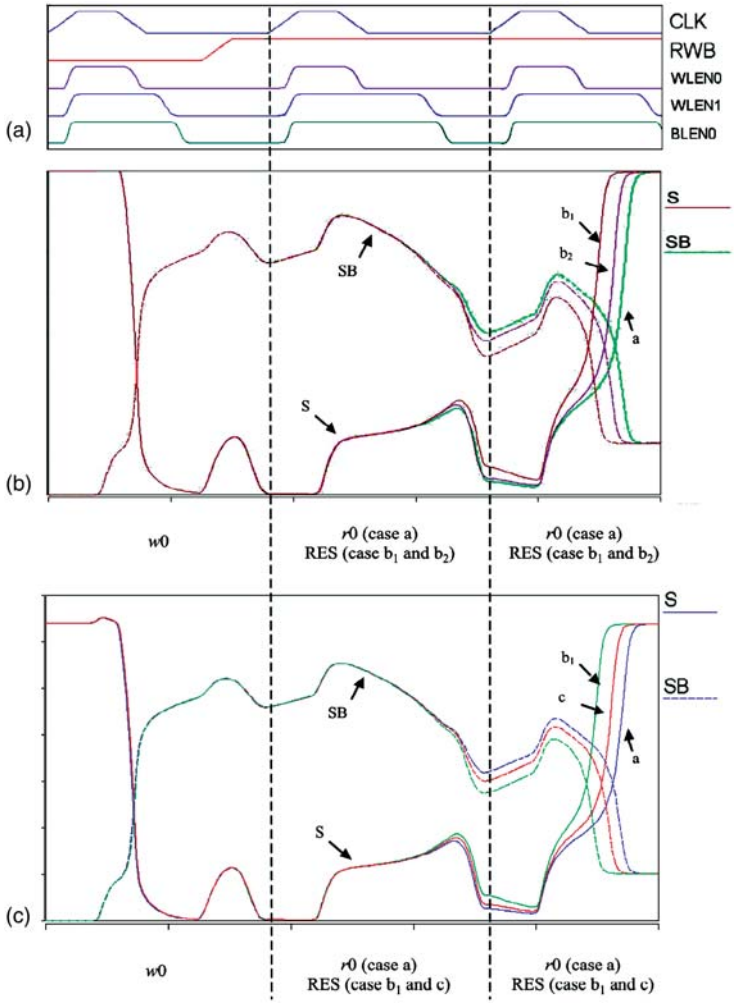


Fig. 2.9 Waveforms of simulations of core-cell affected by defect Df4 = 1.4 MΩ

value of defect Df4. The results referred to case b1 are summarized in the graph in Fig. 2.10. These results are very similar to those shown in Fig. 2.6 that refers to the 'read(s) after write' method. The analysis of the two graphs of Figs. 2.6 and 2.10 also confirms that the sensitization on the faulty core-cell made with RESs is more effective than the one made by actual read operations.

In order to better show the higher effectiveness of RESs for sensitization of dRDF, Table 2.3 gives the results of both read-after-write and RES sensitizations.

In Table 2.3, the values represent the resistive values of defect Df4 that lead to dRDFs. These values are the minimal ones that can be sensitized by 'read(s) after

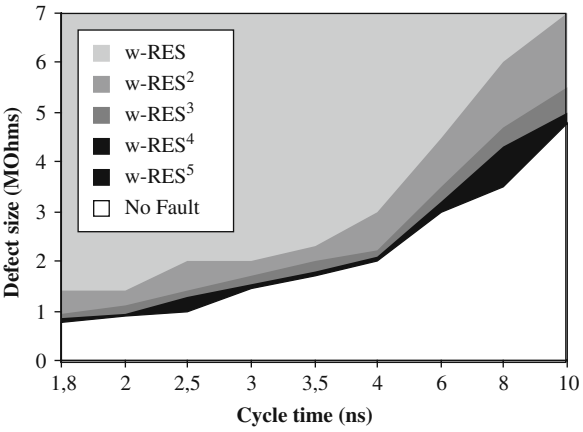


Fig. 2.10 Fault detection as a function of the cycle time, defect size, and RES (process corner typical, $T = 125°C$, $V = 1.6$ V)

Table 2.3 Comparison between 'read(s) after write' and 'RES(s) after write' sensitizations and core-cell affected by defect Df4

Cycle time (ns)		1.8	2	2.5	3	3.5	4	6	8	10
Minimal resistance size, MOhm	w–r	1.7	2	2	3	3.5	4	5.5	8	9.5
	w–RES	1.4	1.4	2	2	2.3	3	4.5	6	7
	w–r^2	1.2	1.2	1.7	2	2.5	3	4	6	7
	w–RES^2	0.95	1.1	1.4	1.7	2	2.2	3.5	4.7	5.5
	w–r^3	0.95	1.1	1.4	1.7	2	2.5	3.5	5	6
	w–RES^3	0.85	0.95	1.3	1.55	1.8	2.1	3.2	4.3	5
	w–r^4	0.9	1	1.3	1.55	1.9	2.2	3.3	4.4	5.5
	w–RES^4	0.8	0.9	1.22	1.5	1.75	2	3	3.5	4.8
	w–r	0.85	1	1.25	1.5	1.9	2.2	3.3	4.3	5.5
	w–RES^5	0.8	0.9	1.22	1.5	1.75	2	3	3.5	4.8

write' (wxrx^M) or 'RESs after write' (wxRES^M) sequences for different values of the clock period. For example, for a clock period of 4 ns, the sequence wxrx^3 sensitizes a dRDF with a minimal resistive value Df4 = 2.5 MΩ, while in same conditions, wxRES^3 allows to sensitize dRDF due to a Df4 = 2.1 MΩ. The higher efficiency of the sensitization patterns with RESs is true for all combinations of clock period and number of operations.

2.3.3 March Test Solutions Detecting dRDFs

This sub-section shows how the experimental results presented above can be used to generate efficient test algorithms for dRDF detection. Among the various types

of test algorithm, March tests allow to reach a good effectiveness with a small complexity. The March test proposed in Dilillo et al. (2005b) is generated on the basis of the following requirements:

a. The March elements have to be performed on the memory array with addressing order *word line after word line*. This is necessary to maximize the number of consecutive RESs, because the RESs are produced only by operating on the core-cells of the same row. For example, let us consider again the SRAM architecture discussed previously. The read and write operations of the March elements have to be operated first on all the 512 core-cells of the first word line, then on the 512 core-cells of the second word line, and so on.
b. The elements of the March test have to include $w0$ operations necessary for sensitization and $r0$ necessary for observation.
c. Additional elements with $w1$ and $r1$ are needed to detect similar faults generated by resistive-open defects placed symmetrically with respect to defect Df4, i.e., in the pull-up of the second inverter composing the core-cell (see Fig. 2.4).
d. All the elements, in particular the sensitization ones, need to be performed in $\Uparrow$ and $\Downarrow$ addressing order.

In order to justify the last requirement, let us consider again the SRAM architecture introduced above. If the faulty core-cell is $C_{i,\,0}$, the first core-cell of the ith row, a March element like $\Uparrow w0$ operates a $w0$ operation on this core-cell and is immediately followed by $w0$ operations performed on the following 511 core-cells of the same row. These $w0$ operations imply 511 RESs on the faulty core-cell. If the faulty core-cell is the second one of the row, $C_{i,\,1}$, the same March element $\Uparrow w0$ involves 510 RESs on the faulty core-cell. In case the defective core-cell is the last of its row, the element $\Uparrow w0$ does not involve any RES on the defective core-cell after the $w0$ operation. The introduction of a $\Downarrow$ element allows the sensitization phase to be also performed with the opposite addressing order of access on the row. In this condition, the core-cells that endure the maximum number of RESs are those placed in the extremes of the row, while those placed in the middle of the row undergo the smallest number of RESs, i.e., $512/2 = 256$ of RESs. In general, if *nb_cell* is the number of core-cells of each row and *nb_op* the number of operations (read/write) of the March element (e.g., $\Uparrow w0 \rightarrow nb\_op = 1$; $\Uparrow r1w0 \rightarrow nb\_op = 2$), the maximum number of RESs that a core-cell endures is

$$\mathrm{RES}^{\max} = (nb\_cell - 1) \cdot nb\_op$$

and the minimum one is

$$\mathrm{RES}^{\min} = (nb\_cell \cdot nb\_op) \,/\, 2$$

This is graphically illustrated in Fig. 2.11, where the color of core-cells is darker when they endure a higher number of RESs.

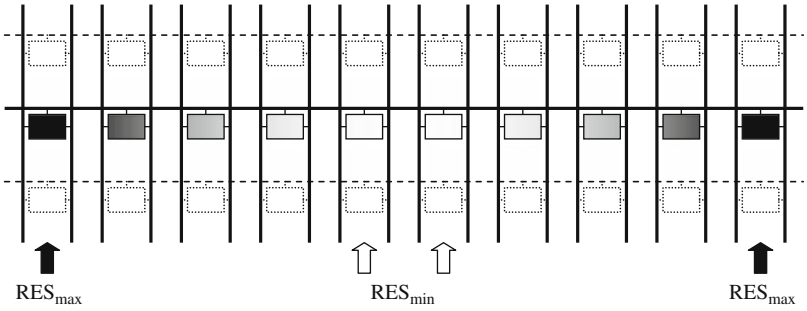


Fig. 2.11 Distribution of RESs on a core-cell row with the modified March C-

There are various March tests, which already embed most of the requirements described above and that can be modified with the objective to detect dRDFs. For this purpose, the well-known March C- (Marinescu 1982) is considered (Dilillo et al. 2005b). This is a 10 N linear test, which is effective to detect stuck-at, transition, and 2-coupling faults and that normally covers 0% of dRDFs (Hamdioui et al. 2002). March C- has the structure shown in Fig. 2.12.

$$\{\Updownarrow (w0)\ \Uparrow(r0, w1)\ \Uparrow(r1, w0)\ \Downarrow(r0, w1)\ \Downarrow (r1, w0)\ \Updownarrow (r0)\ \}$$
$$\text{M0} \qquad \text{M1} \qquad \text{M2} \qquad \text{M3} \qquad \text{M4} \qquad \text{M5}$$

Fig. 2.12 March C- structure

The first five elements (M0–M4) could be effective for dRDF sensitization because they contain the $w0$ or $w1$ operation. In these elements, the read operations, useful for the observation phase, also contribute to the sensitization phase. Both $\Uparrow$ and $\Downarrow$ addressing orders are operated allowing a good sensitization for all the core-cells.

The modification, which makes March C- able to detect dRDFs, consists in the use of the particular address sequence *word line after word line*. Due to the first of the six degrees of freedom (DOF) of March tests (see Section 1.3.4), this modification does not change the capability of March C- to detect the former targeted faults.

The efficiency of the modified March C- can be evaluated considering a given SRAM architecture (Dilillo et al. 2005b). Let us assume that the defective core-cell is $C_{i,0}$, the first one of row i. Element M2 operates a $w0$ on $C_{i,0}$, and just after the couple of ($r1$,$w0$) operations on the following 511 core-cells of the same row i, resulting in $2 \cdot 511 = 1022$ RESs on the defective core-cell. The same happens if the defective core-cell is $C_{i,511}$, the last one of row i, and the element M4, which is M2 with inversed addressing order, is operated: on core-cell $C_{i,511}$, a $w0$ operation

is acted followed by 1022 RESs. Core-cells $C_{i,0}$ and $C_{i,511}$ endure the maximum number of RESs because they are placed in the extremes of their row. The core-cells placed in the middle of the row are the less stressed with $2 \cdot (512/2) = 512$ RESs. Note that elements M1, as its homologue M3, allows the test of similar faults due to resistive-open defects symmetrically placed with respect to Df4.

This modified version of March C- presents many advantages such as its linear complexity and the reuse of an already existing test algorithm, but the main benefit is its high efficiency to detect dRDFs when compared with existing 'read(s) after write' (RAW) test solutions. In fact, in order to reach the same effectiveness, a RAW test should include a very large number of consecutive read operations, leading to a dramatic increase of its complexity.

The modified March C- is also an efficient test for the coverage of all the static fault models that have been identified in a core-cell affected by resistive-open defects. In fact, the modification of March C- makes it able to detect dRDFs, but, due to the first of the six degrees of freedom (see Section 1.3.4) of March tests, it does not change the capability of March C- to detect the former target faults. Among these faults there are the static faults TFs and IRFs that are induced by defects Df1, Df5, and Df6.

As described in Borri et al. (2005), the best sequence to detect the static faults RDFs and DRDFs (involved by defect Df2 and Df3) is $0w1r1$. With this sequence, a defective core-cell swaps its value from the logic '1' stored to logic '0.' Another $r0$ is needed to observe the fault. March C- does not apparently embed the test pattern $0w1r1$, and other algorithms that contain it should be more suitable, e.g., March Y (Van de Goor 1998). As done for the test of dRDF due to defect Df4, the effect of RES can be used making the modified March C- able to produce the needed sequence to exhaustively cover all core-cell static faults.

Element M1 ($\Uparrow$ $r0$, $w1$) operates a $r0$ ('0' was previously stored) followed by a $w1$ operation. The $r1$ stimulus, useful to complete the sensitization of a RDF or a DRDF, can be replaced by the first of the RESs produced by the modified March C-, when M_1 is operated on the following core-cell of the row. This RES warrants the swap of the core-cell for the sensitization as well as an actual read operation. In Dilillo et al. (2005b), the authors verified this fact with SPICE simulations. Note that after the core-cell swap other RESs do not lead to a masking effect with further swaps of the core-cell. In Fig. 2.13, the waveform on the top represents the clock signal (CLK) while the other ones represent the voltage level of node S (see scheme in Fig. 2.1) of a defective core-cell for three values of resistance of defect Df2. For Df2 = 13800 Ω, there is no swap of the core-cell content. Otherwise, for slightly higher values of resistance, i.e., 14100 and 14200 Ω, node S switches from logic '1' to logic '0' due to the first RES and, as expected, there is no masking effect with other core-cell swaps with the following RESs. The observation of the RDF (or similarly of the DRDF) is made by the $r1$ operation of the following March element M2. Equivalent experimental results are obtained simulating a core-cell affected by defect Df3. In conclusion, with the action of one RES, element M1 allows the sensitization of RDF and DRDF, induced by defects Df2 and Df3.

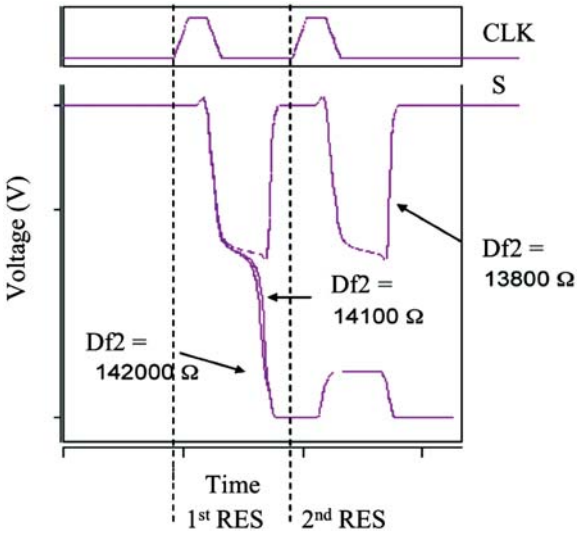


Fig. 2.13 Waveforms of simulations (defect Df2)

2.4 Analysis and Test of dDRF

Some resistive-open defects, which are placed in the refreshment loop of the core-cell, may cause Data Retention Faults (DRFs). There are different forms of DRF depending on the location of the defect and its resistance value. In particular, two sub-cases of dDRFs can be identified and their sensitization can be done by using the RES principle, as seen before for the dRDFs, and by emphasizing natural noise phenomena in SRAMs such as the ground bounce (Dilillo et al. 2005a).

2.4.1 Functional Fault Modeling of dDRF

The functional fault model of dDRF is composed of the following FP set:

$$FP_{dDRF} = <ad1(yop_0op_1...op_{M-1}), ad2(x)/ad2(\bar{x})/->$$

where

- $ad1 \neq ad2$ and *ad1*, *ad2* belong to the same row;
- $op_i \in \{w_d, r_k\}$, $0 \leq i < M$ and $M \geq 2$;
- y, x, $\bar{x}$, d, k $\in \{0,1\}$ and x $\neq \bar{x}$.T.

The FP_{dDRF} allows to formalize the following faulty behavior: a core-cell *ad2* is set to a logic value x, then on a different core-cell *ad1* initialized with a logic

value y, a sequence of at least two operations ($M \geq 2$) are applied. Since *ad1* and *ad2* belong to the same row, the M operations applied on the core-cell *ad1* force a RES^M on the core-cell *ad2* that causes the inversion of the logic value x stored on *ad2* (from x to $\bar{x}$).

The faulty behavior induced by a dDRF is analyzed in Section 2.4.2 through electrical simulations.

2.4.2 Experiments

For the study of dynamic DRFs, it is appropriate to first introduce static DRFs that occur when a memory core-cell loses its previously stored logic value after a certain period of time during which it has not been accessed (Van de Goor 1998). In general, this kind of fault is the consequence of resistive-open defects in SRAM core-cells, in particular in the refreshment loop (INV1 + INV2). Figure 2.14 depicts the scheme of a standard six-transistor core-cell with the three resistive-open defects commonly causing DRFs (Van de Goor 1998, Wang et al. 2003). These defects are referred to as Df2, Df3, and Df4, in order to be consistent with the notation introduced in Fig. 2.4.

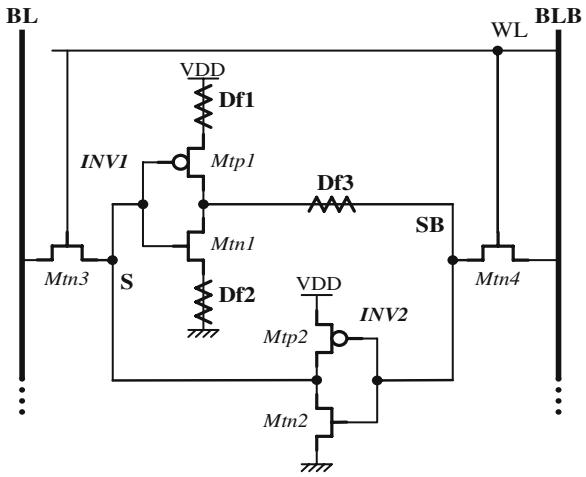


Fig. 2.14 Resistive-open defects injected into the memory core-cell causing DRFs

In order to explain the mechanism of DRFs, let us consider a core-cell with a resistive-open defect in the pull-up of the inverter INV1. This defect, referred to as Df4, limits the current flow between VDD and node SB. In Section 2.3, this defect has been presented as cause of Read Destructive Faults (RDFs), or dynamic Read Destructive Faults (dRDFs), with resistance values in a certain range (low). When Df4 has a very high resistance value, larger than 100 MΩ (for 130 nm technology), it almost behaves like a pure open circuit and involves DRFs.

When a logic '0' is stored in the core-cell ($w0$), bit line BL discharges node S to '0,' while BLB charges node SB to logic '1.' At the end of the $w0$ operation, the two pass-transistors Mtn3 and Mtn4 are OFF and the core-cell is isolated from the bit lines. At this moment, node S is active low due to the pull-down of inverter INV2 (see Fig. 2.14), while node SB is floating high because the pull-up of INV2 does not work properly due to the presence of defect Df4. In these conditions, if the leakage current that discharges node SB (especially through Mtn1 and Mtn4) is higher than the current that passes through the faulty pull-up of INV1, the voltage value of node SB decreases from the initial value VDD. When the voltage level of node SB reaches the threshold value of VDD/2, the inverter INV2 switches. Immediately after, the inverter INV1 switches, too; thus a faulty swap of the core-cell content occurs.

This phenomenon has been studied in Dilillo et al. (2005a) on an industrial 130 nm embedded SRAM, considering the case of a core-cell affected by defect Df4, with the following operating conditions:

Process corner:	fast
Supply voltage:	1.6 V
Temperature:	120°C
Resistive-open defect:	Df4 = 100 MΩ

The evolution of the phenomenon of DRF, described above, is clearly shown by the waveforms in Fig. 2.15 , referred to the voltage levels of nodes S and SB.

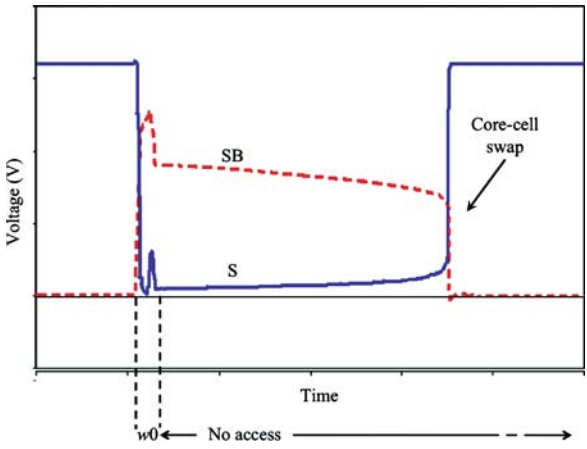


Fig. 2.15 Data retention fault caused by high resistive defect Df4

Immediately after the $w0$ operation, node S is at logic '0,' while node SB is at logic '1' but with a voltage level lower than VDD due to the faulty pull-up of INV1. For the same lack of pull-up, node SB slowly loses the stored charge because of the leakage currents. With the decrease of the voltage of node SB, also INV2 begins to work not perfectly, in particular transistor Mtn2 starts to conduct less than in normal condition while transistor Mtp2 (supposed to be OFF) begins to conduct. Consequently, the voltage of node S slowly grows. When node SB reaches the threshold of VDD/2, INV2 switches and the core-cell content swaps. Note that,

in this experiment, the particular simulation parameters (high temperature 125°C and the fast process corner) maximize the occurrence of DRFs.

In order to sensitize this kind of DRF, it can be sufficient to leave the memory in hold mode (no operation acted) for a certain time (Van de Goor 1998), waiting for the data loss. But there is another more efficient way. In fact, the same high resistive-open defects that cause DRFs are also the source of Read Destructive Faults (RDFs) or Deceptive Read Destructive Faults (DRDFs) (Hamdioui and Van de Goor 2000). For this reason, for the detection of DRFs, we can use the read after write pattern (March RAW (Hamdioui et al. 2002)) that does not employ delays. In practice, if immediately after a write operation (e.g., $w0$) a read operation is acted (e.g., $r0$), the fault is sensitized with a swap of the core-cell content. After the $w0$ operation, in presence of defect Df4, the information is stored correctly, but while node S is active low, node SB is floating at logic '1.' The following read operation connects the two nodes, S and SB, to the two bit lines pre-charged at VDD. Consequently, bit line BL pulls-up node S and it partially activates transistor Mtn1. Transistor Mtn1 is not balanced by transistor Mtp1 and discharges node SB, causing the switch of inverter INV2 and, consequently, the swap of the core-cell.

In the case, the current that passes through the resistive-open defects Df2, Df3, and Df4 is larger than the leakage current in core-cell node SB (i.e., in presence of low resistive defects), data loss may still occur, but the behavior of the faulty core-cell is more complex. In these conditions the DRFs are dynamic. This case, discussed in detail in Dilillo et al. (2005a), is reported in the next two sub-sections. Section 2.4.2.1 studies the case of defect Df4 that does not behave like a quite pure open circuit (Df4 < 100 MΩ). Section 2.4.2.2 studies the case of defects Df2 and Df3 that are far from the condition of pure open circuits, i.e., for resistive values in the order of some kΩ.

2.4.2.1 dDRF Due to Defect Df4

When defect Df4, with resistive values that allow a certain current flow, is injected in a SRAM core-cell, a data loss may occur corresponding to a DRF. In this case, the sensitization of such DRF is different from that presented in the previous section. Let us consider, for example, a core-cell of a 130 nm industrial embedded SRAM affected by the resistive-open defect Df4 with a resistive value lower than 100 MΩ. After a $w0$ operation, the data is stored in the core-cell correctly with node S at logic '0' and node SB at logic '1.' Normally, if no operation is performed on the defective core-cell, it continues to remain stable along the time, because the leakage current of node SB is lower than the current provided by the pull-up of INV1, even in presence of defect Df4, because although no operation is performed directly on the defective core-cell it may be subject to indirect stimulation (RES) and lose the stored data. As shown before, when a core-cell is accessed for a read/write operation, the word line signal activates the pass-transistors of all the core-cells belonging to the same row. While the operation is done on a selected core-cell, all the other core-cells of the same word line are connected with their bit lines kept at VDD by the pre-charge circuits. Section 2.3.2 discussed on how the stress on indirectly selected core-cells

(Read Equivalent Stress, RES) is very similar to the stress produced by an actual read operation. In non-defective core-cells, the RES does not lead to data loss, but if applied on a core-cell affected by Df4 it produces a degradation of the voltage level of core-cell nodes. In fact, node SB of the defective core-cell loses its charge and its voltage level drops. For example, Fig. 2.16 shows the waveforms for a core-cell with defect Df4 and the following operating conditions:

- *Process corner:* typical
- *Supply voltage:* 1.6 V
- *Temperature:* 27°C
- *Resistive-open defect:* Df4 = 15 MΩ

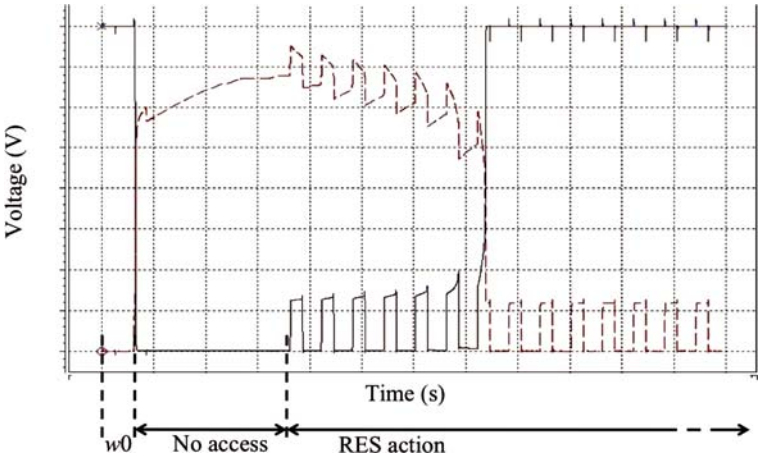


Fig. 2.16 dDRF caused by low resistive Df4

This value of resistance may appear large, but is very little in comparison to the values that involve classical DRFs with this process corner and temperature.

The waveforms of Fig. 2.16 show that after a *w*0 operation the logic value is correctly stored in the core-cell: node S is at '0' and node SB is at logic '1.' However, due to the faulty pull-up of INV1, node SB has not reached VDD perfectly. This voltage level is reached during the following four clock cycles, when the core-cell is accessed neither directly with read/write operations nor indirectly by RESs. At this moment, the voltage levels of S and SB seem not to be influenced by leakage currents as in the case of very high resistive Df4 that leads to static DRF. After this no-operation period, the defective core-cell has been stimulated indirectly by RESs, i.e., by acting read/write operations on other core-cells placed on the same row. The consecutive RESs produce the progressive degradation of the voltage of node SB. This node, after seven RESs, reaches the threshold of VDD/2 and the core-cell swaps. Consequently, the core-cell presents a DRF because with the normal use of the memory it loses its stored value.

2.4.2.2 dDRF Due to Defects Df2 and Df3

In some ranges of resistance values for defects Df2 and Df3, a peculiar behavior of the defective core-cell, leading to data loss, can be observed (Dilillo et al. 2005a). For the considered 130 nm memory, these ranges of resistance are the following ones:

- **Df2** < 11.8 kΩ
- **Df3** < 5.3 kΩ

In order to explain this case, the experiments detailed in Dilillo et al. (2005a) are reported here with a simulation of a 130 nm core-cell, in which defect Df2 is inserted with the following operating conditions:

- *Process corner:* typical
- *Supply voltage:* 1.6 V
- *Temperature:* 27°C
- *Resistive-open defect:* Df2 = 11.7 kΩ

A meaningful experimental result is given in Fig. 2.17. First, a *w*1 operation is correctly acted: node S is a logic '1' (VDD), and node SB is at logic '0' (GND). After, a sequence of four clock cycles follows, with no degradation of the voltage level of the two core-cell nodes (*no access* region in Fig. 2.17). With the following clock cycles, the defective core-cell endures the stress of a sequence of RESs. In the previous example, this stress was sufficient to produce the progressive degradation

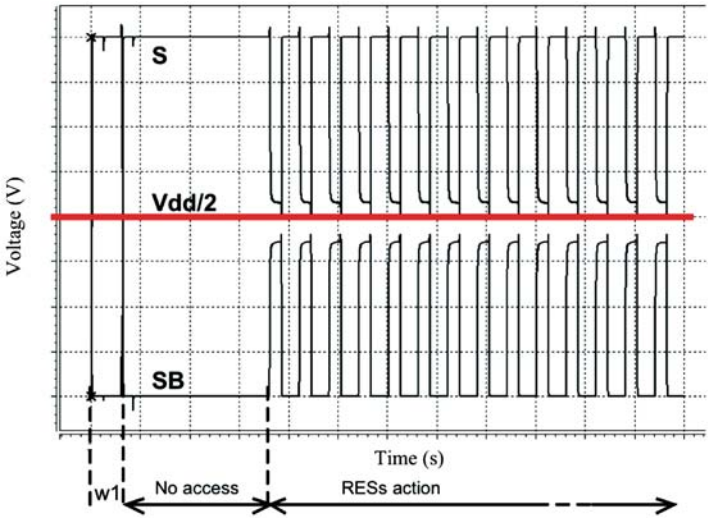


Fig. 2.17 Core-cell instability that may produce Data Retention Faults. Df2 = 11.7 kΩ

of SB voltage level and, after a certain number of RESs, the swap of the core-cell content. In this case, during the action of the RESs on the core-cell, the voltage level of nodes S and SB only gets close to the threshold voltage VDD/2. This phenomenon appears at each cycle, but there is no actual faulty swap of the core-cell content, whatever the number of RESs in the sequence. The same considerations and analysis are also valid for defect Df3.

Despite the fact that the core-cell does not swap in this experiment, the detection of defects Df2 and Df3 is important for two reasons. The voltage level reached by node S and node SB are very close to the commutation threshold: about 10 mV in the example of Fig. 2.17. An SRAM is expected to retain data for a long time, sometimes for years (Van de Goor 1998), and a core-cell that presents the behavior shown in Fig. 2.17 is very weak during all normal operations that involve RESs. If this weakness is coupled with noise phenomena, like the ground bounce, the core-cell that may have passed the common tests for DRF may swap losing its logic state during the normal operation of the memory. Moreover, even if during the RESs action the voltage levels of node S and SB are not enough close to the VDD/2 threshold to induce the swap, the device may face a premature aging. In fact, the presence of a resistive-open defect may induce a local increment of the heat due to the Joule effect and a consequent increase of the temperature. In this condition, the phenomenon of electromigration appears, and the resistive-open defect grows till to produce the malfunction of the core-cell. The cycle of the acceleration of electromigration is shown in Fig. 2.18.

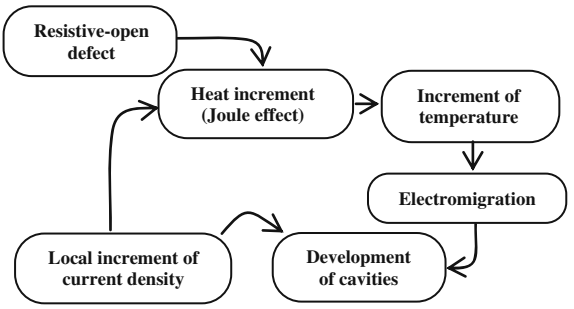


Fig. 2.18 The acceleration electromigration cycle

2.4.3 March Test Solution Detecting dDRFs

Concerning the DRF caused by high resistive defect Df4, some algorithms have been proposed in literature. In particular, in Van de Goor (1998), two test solutions are proposed: IFA-9 and IFA-13 which are shown in Fig. 2.19.

These two algorithms cover many fault models such as stuck-at faults, unlinked coupling faults, transition faults, and others. Their capability to detect DRFs is

Fig. 2.19 IFA-9 and IFA-13 test procedure

$\{\Uparrow (w0)\ \Uparrow(r0, w1)\ \Uparrow(r1, w0)\ \Downarrow(r0, w1)$
$\Downarrow (r1, w0)$ Del $\Uparrow(r0, w1)$ Del $\Uparrow(r1)\}$
IFA-9

$\{\Uparrow (w0)\ \Uparrow(r0, w1, r1)\ \Uparrow(r1, w0, r0)\ \Downarrow(r0, w1, r1)$
$\Downarrow (r1, w0, r0)$ Del $\Uparrow(r0, w1)$ Del $\Uparrow(r1)\}$
IFA-13

related to the presence of delay periods (referred as Del) inserted after March elements that leave respectively logic '0' and logic '1' stored in all the core-cells of the array, for the coverage of symmetrically placed resistive-open defects. The duration of the delay is strictly related to the architecture of the memory and the manufacturing technology. The delay time may vary in a range from some 100 ns up to 10 ms.

In the case of defects with low resistance value, the IFA algorithms are not useful, because the delays Del, which were functional to the sensitization of static DRF, now conversely contribute to stabilize the core-cell. This fact can be observed in Fig. 2.16, where, in the *no access* region, the voltage level of node SB grows to reach VDD. In order to detect this kind of DRF, it is useful to produce the highest number of consecutive RESs. In fact, a sequence of consecutive RESs produce a good sensitization of the fault by degrading progressively the voltage of one of two core-cell nodes. An easy way to produce long sequence of RESs is the application of a common March test with the addressing order *word line after word line*.

In order to detect the presence of defects Df2 and Df3, an algorithm has to have the following requirements:

a. Maximize the productions of RESs in sequence.
b. Maximize at same time the noise conditions.

The modified March C-, presented in Section 2.3, produces a large number of consecutive RESs, and thus is valid to cover dDRFs due to defect Df4 and to cover the first requirement for the detection of defects Df2 and Df3. For the second requirement, a dedicated source of noise can be introduced during the application of the algorithm. Another testing way is the application of the algorithm in conditions that maximize the natural sources of noise in SRAMs.

Among the possible noise sources in SRAMs, the ground bounce can be considered (Senthinathan and Prince 1991, Ding and Mazumder 2003). The ground bounce is caused by large instant current, due to the switching of multiple devices, through parasitic inductance at the ground node. Ground bounce is a serious problem especially in semiconductor memories, because of the simultaneous switching of a large number of memory core-cells and sense amplifiers. The combination of

the effects of ground bounce and the presence of defects Df2 and Df3 may easily produce the faulty swap of the core-cell. In fact, when '1' is stored, the voltage level of the node at '0' (node SB) can shift down to some 100 mV (Ding and Mazumder 2003) due to ground bounce. Consequently, defective core-cells with defect Df2 or Df3, which present non-detectable malfunction in normal conditions, may swap in the presence of ground bounce and RESs.

As mentioned above, the ground bounce appears when there is a multiple gate switching. In memories, the devices that may switch are the core-cells, the output circuits, and the addressing circuits, especially those used for the column selection that commonly employ several levels of multiplexers. On the basis of these considerations, an algorithmic procedure that maximizes the detection of defects Df2 and Df3 can be formulated. As done for the detection of dRDFs, existing March test with the addressing order *word line after word line* can be used for the production of RESs in sequence. Moreover, in order to maximize the incidence of ground bounce, the March test must also have the following requirements (Dilillo et al. 2005a):

a. The part of the address concerning the bit line selection has to present the largest possible Hamming distance between each couple of addresses. In other words, this part of the address has to produce the highest possible number of bit commutations. For example, on an 8 bit column address the change between the two addresses 01001110→10110001 leads to 8 commutations, the maximum possible.
b. The stored logic value has to change for each write operation and so for each read operation. This requirement is allowed by the fourth of the six degrees of freedom of March tests (see Section 1.3.4).

In bit-oriented memories, it is easier to operate a large number of switching on the column addressing circuits. In the case of a word-oriented SRAM, the ground bounce phenomenon can also be maximized by provoking the switching of all the output bits. Note that this test solution does not assure 100% coverage, because many factors, as the resistive value of defects, are concerned, but assures the maximization of the testability.

2.5 Impact of Technology Scaling

This section presents a comparison of the effects produced by resistive-open defects in SRAM core-cell with different degrees of integration from 130 nm down to 45 nm (Dilillo et al. 2005c, Dilillo et al. 2008). For this purpose, the results of electrical simulations are reported considering an 8 K × 32 memory organized as an array of 512 word lines × 512 bit lines. The resistance values are chosen in a range between few ohms up to several Mohms to produce a complete view on the studied phenomena. In order to allow a correct comparison between the different technologies,

similar simulation conditions have been considered for all technologies with the typical working parameters:

- *Temperature:* 25°C
- *Process corner:* typical
- *Supply voltage:* (130 nm, 1.5 V), (90 nm, 1.2 V), (65 nm, 1 V), (45 nm 0.9 V)
- *Simulations performed in the 'active mode' (normal functional mode)*

In Fig. 2.20, the graph represents for each technology the minimal value of resistance that leads to malfunctions. Note that the scale of the resistance is logarithmic.

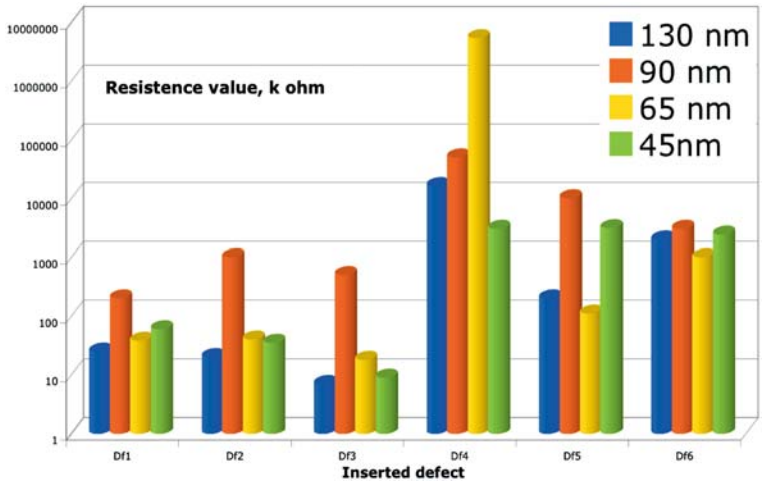


Fig. 2.20 Graph of the minimal resistive values that produce faulty behavior

Before introducing any comment on the simulation results, it is useful to make some considerations. The 130 nm and 90 nm memories are produced for several years and a lot is known about them. When the simulations were performed, the 65 nm and especially 45 nm memories were young technologies, and the accuracy of their model was probably lower than for the other older technologies.

The analysis of the results has shown a crucial difference concerning the robustness of the core-cell in the presence of resistive-open defects. This is related to the capability of the core-cell to keep working correctly even in the presence of resistive-open defects, until a certain threshold value of resistance. The 90 nm technology is generally the most insensitive to the defect injection. In other words, the 90 nm core-cell works correctly for resistive values that conversely induce a faulty behavior in the 130 nm, 65 nm (except for defect Df4) and 45 nm core-cells. Defects Df4 and Df6 apart from the other defects require a resistance value that normally is one order of magnitude higher in the case of the 90 nm core-cell. With the 65 nm technology, the core-cell seems to be particularly insensitive to defect Df4 with faulty behavior appearing starting with a resistive value of 5.3 GΩ.

For defect Df6, the simulations give similar results for all technologies. This factor can be explained by the position of defect Df6 that is placed at the gates of the two transistors of the inverter INV2. No bias current enters in the CMOS transistor gate; thus, independently of the technology, defect Df6 leads to similar effects.

A global analysis of the graph reveals that the results do not present a uniform trend. This factor may be explained by two important arguments. The first one is that the more scaled devices are not simply obtained by shrinking the transistor dimension of the core-cell, even if they belong to the same family. In particular, the more scaled devices present major problems in terms of current leakage, process variations, etc., leading to consequent design choices. On the other hand, different values of supply voltage have been used for the simulations with the aim to respect typical supply voltage of each technology (e.g., 1.5 V for the 130 nm technology and 1.2 V for the 90 nm technology). For the injection of resistive-open defects, the faulty behavior appears more frequently at high voltage: high supply voltage makes the memory surprisingly less stable than lower supply voltage resulting in 130 nm core-cells more sensitive to resistive-open defects.

Furthermore, simulations, performed with different operating conditions in terms of temperature, supply voltage, and process corner, have been proposed in Dilillo et al. (2008) (VTS). The results are summarized in Table 2.4 and concern only the extracted fault models.

Table 2.4 Fault model extraction: simulation results with any parameters

Technology	Df1	Df2	Df3	Df4	Df5	Df6
130 nm	TF	DRDF, RDF, dDRF	DRDF, RDF, dDRF	dRDF, DRF, dDRF	IRF, TF	TF
90 nm	TF	DRDF, RDF, dDRF, dRDF	DRDF, RDF, dDRF, dRDF	dRDF, DRF, DDRF	IRF, TF	TF
65 nm	TF, dRDF	DRDF, dDRDF	DRDF, dDRDF	dRDF	TF, dRDF	TF, dRDF
45 nm	TF	RDF, DRDF	RDF, DRDF, dRDF	RDF, DRDF	TF	TF

In the majority of cases, each defect involves the same fault models in all technologies. Moreover, a visible trend of increased occurrence of dynamic faults in more scaled technologies is evident with the exception of the 45 nm technology. For example, defects Df2 and Df3 lead to static faults in 130 nm core-cells; the same faults are integrated with dynamic faults in the 90 nm technology core-cells. The dynamic nature of the faults brought by Df2 and Df3 is visible in the waveforms in Fig. 2.21 that are referred to two examples of dRDF in 90 nm core-cell respectively affected by defects Df2 and Df3 (Dilillo et al. 2005c). Several read operations ($w1r1^3$ for Df2 and Df3) are necessary to produce the faulty swap of the core-cell content (see encircled transitions in Fig. 2.21).

(a): Df2 – 950 kΩ, mode slow, typ, 1.2 V, 27°C- dRDF
(b): Df3 – 1 MΩ, mode slow, typ, 1.2 V, –40°C- dRDF

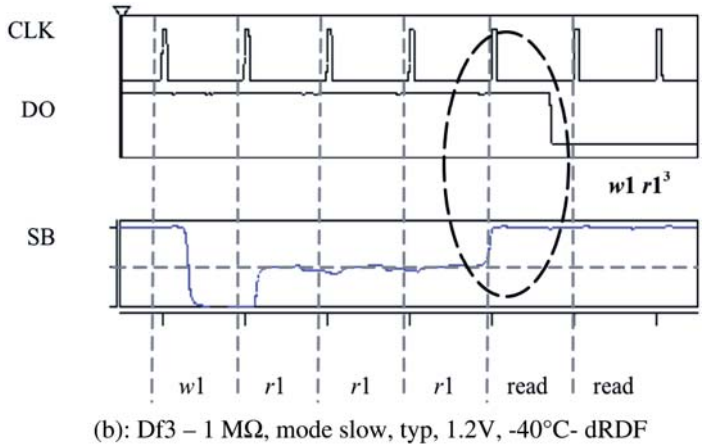


Fig. 2.21 Dynamic faulty behavior induced by Df2 and Df3 in the 90 nm core-cells

Concerning Df2 and Df3, in the case of the 90 nm core-cell, when there is a dRDF, the refreshment of the loop has a degradation that is proportional to the value of the injected resistive-open defect. When a certain resistance value is reached the fault loses its dynamic nature and becomes a simple RDF. One simple read operation is sufficient to produce a faulty swap of the core-cell content. In the case of the 130 nm core-cell, the transition range of resistance value in which there is a dynamic behavior is reduced to a point, named *cut point*. This cut point is at $R = 20$ kΩ for Df2 and $R = 7$ kΩ for Df3. Before this cut point, the value of the injected resistance is not sufficient to induce a faulty behavior. After the cut point all the resistive values lead to a (static) RDF. For larger resistive defects, it is impossible to act correct write operations ($w1$) in the core-cell. This is modeled with a transition fault.

2.6 Conclusion

Electrical simulations show that several static faults such as TFs, IRFs, RDFs, DRDFs, and DRFs may affect SRAM core-cells. These faults are easily covered by common March tests. Other faults that affect core-cells are dynamic, such as dRDFs and dDRFs. For these faults, particular March test solutions are necessary for their coverage. In particular, it has been shown that a core-cell undergoes a stress equivalent to a read operation (RES), when a read/write access is performed on a core-cell of the same row. Considering the RES principle, existing March tests can be modified in order to make them able to detect dRDFs and dDRFs.

The study on the occurrence of static and dynamic faults with technology scaling reveals that the core-cell in 90 nm technology is generally the less sensitive to resistive-open defects, presenting a faulty behavior only for values of resistance in average of one order of magnitude higher than for other technologies. Moreover, dynamic faults are more likely to occur in more scaled technologies.

Chapter 3
Resistive-Open Defects in Pre-charge Circuits

3.1 The SRAM Pre-charge Circuit

In SRAMs, for each column of the core-cell array, there are a couple of bit lines, BL and BLB, which are connected to a pre-charge circuit. The function of this circuit is the setting and equalization at a certain voltage level (often VDD) of the two bit lines. The pre-charge is active as long as the column is not selected and its action is particularly important before the read access. For the read operation, the two bit lines, connected to the selected core-cell, have to be perfectly equalized. A typical pre-charge circuit is depicted in Fig. 3.1. It is composed of two PMOS transistors that connect the bit lines to VDD.

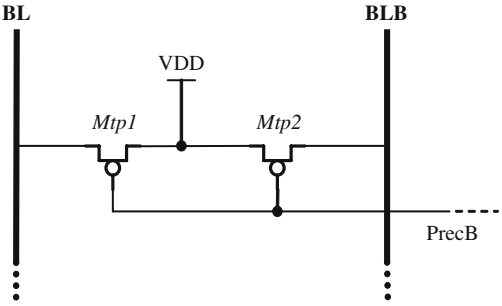


Fig. 3.1 Two transistor pre-charge circuit

As mentioned above, the memory array contains a pre-charge circuit for each couple of BL/BLB, as it can be observed in Fig. 3.2 that depicts a portion of an SRAM array.

When a read operation is acted on a certain core-cell, the pre-charge circuit is turned OFF in the column of the selected core-cell. Thus, just before the core-cell selection, the bit lines have been charged and are floating at VDD. The read operation begins when the Word Line signal (WL_i) allows the connection of the core-cell with its two bit lines. One node of the core-cell is at logic '1' (VDD), and the other is at logic '0' (0 V). During the read operation, one of the two bit lines is partially discharged and reaches the voltage level VDD – ΔBL. A sense amplifier detects the

A. Bosio et al., *Advanced Test Methods for SRAMs*,
DOI 10.1007/978-1-4419-0938-1_3, © Springer Science+Business Media, LLC 2010

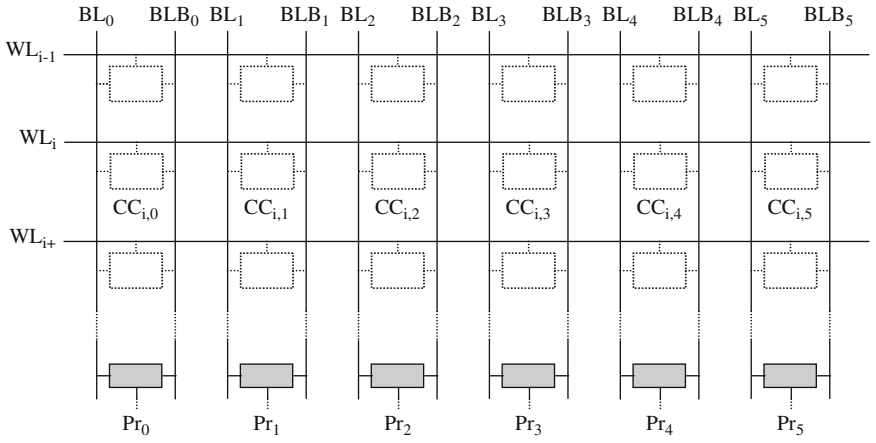


Fig. 3.2 A portion of an SRAM cell array with pre-charge circuits

differential voltage between the two bit lines and gives the output value. The read value is a logic '0' when bit line BL is partially discharged (at VDD – ΔBL) and bit line BLB remains at VDD; a logic '1' is read in the opposite case.

It is crucial that before any read operation the two bit lines connected to the core-cell are charged at VDD and the equalization of the two bit lines is perfectly achieved. Any voltage difference between the two bit lines before the read operation may lead to an incorrect read value, because the read operation itself is based on the amplification of ΔBL ($= V_{BL}–V_{BLB}$) through the sense amplifier. For this reason, in modern SRAMs, the pre-charge circuit is composed not only of two transistors that pull up BL and BLB, but also of a third transistor that ensures the perfect equalization of the two bit lines. The three transistor configuration is depicted in Fig. 3.3.

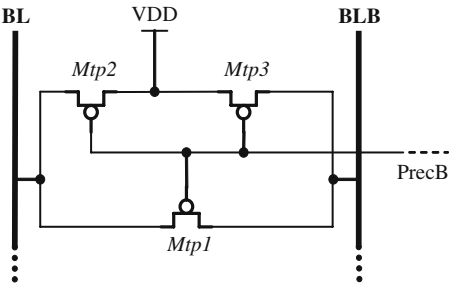


Fig. 3.3 Three transistor pre-charge circuit

In this pre-charge circuit configuration, the three PMOS transistors are driven by a single command signal PrecB. Transistors Mtp2 and Mtp3 connect the bit lines to VDD for the pull-up. Transistor Mtp1 connects the two bit lines for their equalization. In many cases, when one of the two bit lines is already at VDD, Mtp1 helps

the pull-up of the other bit line. In the following, the additional transistor Mtp1 will be proven to ensure a lower sensitivity of the pre-charge circuit in presence of resistive-open defects.

3.2 Analysis of Resistive-Open Defects in the Pre-charge Circuit

The study on the influence of resistive-open defects in the pre-charge circuit is currently made through defect injection in the circuit itself (Dilillo et al. 2005d). For the sake of simplicity, only one defect for each analysis can be considered, because the occurrence of multiple defects has low probability in such a small circuit composed of only three transistors.

3.2.1 Defect Location

As shown in Fig. 3.4, five resistive-open defects (Df1 to Df5) can be placed in different locations of the analyzed circuit (Dilillo et al. 2005d). From these five locations, the remaining defect locations can be deduced by considering the symmetry of the pre-charge circuit structure. In particular, defects Df1, Df3, and Df4 are placed in the right part of the circuit and have their symmetric on the left part of the circuit. Defects Df2 and Df5 do not have any symmetric defect in other part of the pre-charge circuit because they are placed in the middle of the structure.

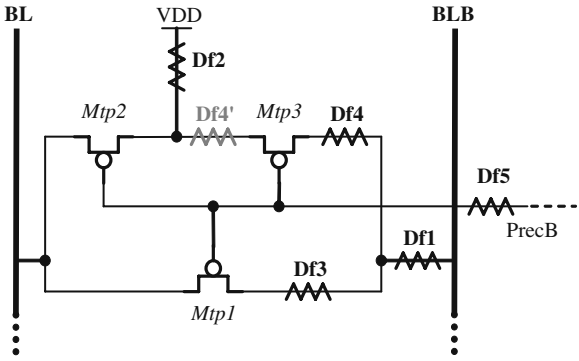


Fig. 3.4 Resistive-open defects injected into the pre-charge circuit

On this basis, the resistive-open defects injected in the pre-charge circuit can be grouped in two main categories:

Asymmetric defects: This group includes defects Df1, Df3, and Df4 that contribute to an asymmetric pre-charge, introducing a delay in the pull-up of BLB.
Symmetric defects: This group includes defects Df2 and Df5 that produce a delay in the pull-up action of both bit lines.

As mentioned above, the asymmetric defects produce similar effects if they are placed on the left side of the circuit (the BL side). Similarly defect Df4′ is not considered in the simulations because it produces effects very similar to defect Df4. In fact, they are both placed in the same current path that feeds bit line BLB.

3.2.2 Defect Incidence Analysis

A qualitative analysis of each defect on the function of the pre-charge circuit, reported in Dilillo et al. (2005d), is detailed below:

- *Defect Df1:* This defect is asymmetric and represents the most prominent cause of malfunction because its action not only prevents the correct pull-up of bit line BLB, but also blocks the equalization effect of transistor Mtp1. The presence of defect Df1 contributes to accelerate the asymmetric charge of bit line BL through transistor Mtp1.
- *Defect Df2:* This defect is symmetric and prevents the correct charge flow from VDD to the two bit lines. Defect Df2 may cause a faulty behavior when, before the pre-charge action, a bit line is at VDD and the other one is at 0 V, e.g., $V_{\mathrm{BL}} =$ VDD and $V_{\mathrm{BLB}} = 0$ V. In this case, there is a delay in the charge flow from VDD to bit line BLB. At the end of the pre-charge phase, there is a certain ΔBL that influences the following read operation. The effect of the defect Df2 on the read operation is strongly reduced by the presence of transistor Mtp1. In fact, even if the VDD level is not reached by both bit lines, transistor Mtp1 ensures their equalization, as needed for a correct read.
- *Defect Df3:* This defect is asymmetric and influences both pre-charge and equalization. When the initial state is $V_{\mathrm{BL}} =$ VDD and $V_{\mathrm{BLB}} = 0$ V and the pre-charge is acted, transistor Mtp3 only may not be sufficient to pull up bit line BLB to VDD. In a defect free circuit, the charge provided by transistor Mtp2 helps the pull-up of bit line BLB through transistor Mtp1 (bit line BL is already at VDD). Moreover, when the two transistors Mtp2 and Mtp3 are activated, they contribute to the equalization of the bit lines. This fact explains that the memory can behave correctly even in presence of defect Df3. This situation can be different in the case of operation at high frequency or in the presence of small size transistors. In these cases, the pull-up time can be not large enough to correctly set both bit lines at the same voltage level.
- *Defect Df4:* This defect is asymmetric and disturbs the correct pull-up of bit line BLB. The effect of Df4 is partially reduced by the presence of transistor Mtp1 that contributes to pull-up bit line BLB, taking the charge from both transistor Mtp2 and bit line BL, already at VDD. In practice, the two bit lines are charged by the sole transistor Mtp2, but bit line BL can be more charged than bit line BLB because the current flows only through one transistor (Mtp2) and not two (Mtp2 and Mtp1). The comments given for defect Df3, in terms of frequency and transistor size, are also valid for defect Df4.

– *Defect Df5:* This defect is symmetric and implies a faulty behavior of the memory. This defect may represent the resistive effect of the connection of the pre-charge command signal. The main difference with defect Df2 is that, in this case, there is a double delay action: the delay in the pull-up acted by transistors Mtp2 and Mtp3 and the delay in the equalization done by transistor Mtp1. The faulty behavior appears for high values of resistance because defect Df5 is located at the gates of the pre-charge transistors. No bias current enters in the MOS transistor gate, and hence the defect has to be highly resistive in order to generate a sufficient delay leading to a faulty behavior.

Table 3.1 presents a summary of the fault models identified for each injected resistive defect.

Table 3.1 Fault models identified for each injected defect

Defect	Fault model
Df1	URWF, URRF
Df2	–
Df3	–
Df4	–
Df5	URWF, URRF

All malfunctions caused by the resistive-open defects in the pre-charge circuit can be modeled by two dynamic-fault models, the URRF and the URWF that are defined as follows:

Un-Restored Write Fault (URWF): In a certain column, the pull-up and equalization of the two bit lines is not completely acted after a write operation. Consequently, the following read operation of an opposite data in a core-cell of the same column is not correctly acted.
Un-Restored Read Fault (URRF): In a certain column, the pull-up and equalization of the two bit lines is not completely acted after a read operation. Consequently, the following read operation of an opposite data in a core-cell of the same column is not correctly acted.

3.2.3 Simulation Set-Up and Results

The previous analyses have been confirmed by SPICE simulations of the pre-charge circuit in Dilillo et al. (2005d), performed on a 130 nm 8 K $\times$ 32 SRAM organized as a 512 $\times$ 512 array with the following operating conditions:

- *Temperature:* 25°C
- *Process corner:* Typical
- *Supply voltage:* 1.5 V
- *Clock cycle time:* 3 ns
- *Resistance values:* 1 Ω up to some G Ω.

Table 3.2 Summary of simulations with related fault models and corresponding minimum detected resistance

Defect	Faulty effect and minimum resistance
Df1	URWF (2 kΩ)/URRF (4 kΩ)
Df2	Low BL voltage, but no fault detected
Df3	Weak ΔBL in the read operation, but no fault detected
Df4	Weak ΔBL in the read operation, but no fault detected
Df5	URWF (35 kΩ)/URRF (45 kΩ)

The results of the parametric simulations, employing the test patterns for URWF and URRF, are summarized in Table 3.2. The first column lists the resistive-open defects; the second one gives the faulty effects and the minimum resistance values that involve the faulty behavior.

The analysis of Table 3.2 shows that only defects Df1 and Df5 lead to actual malfunctions (detectable faults). These defects produce a faulty behavior from a certain resistance value that is equal to 2 kΩ for defect Df1 and 35 kΩ for defect Df5.

Defect Df2 involves an incorrect pull-up of the bit lines, but does not lead to an actual faulty read operation. Although VDD is not reached during the pre-charge phase, the redundancy of the pre-charge circuit ensures a good equalization of the bit lines, which is a sufficient condition to achieve a successful read operation.

In the presence of defects Df3 or Df4, the bit lines are not equalized at the end of the pre-charge phase, but the resulting ΔBL is not large enough to imply a faulty behavior of the memory.

Globally, these results show that the three-transistor pre-charge structure (with the redundant transistor Mtp1) presents a certain hardware redundancy that makes this circuit less sensitive to resistive-open defects.

3.3 Analysis and Test of URRF and URWF

The simulations confirm that an incorrect pre-charge/equalization of the bit lines after a write or read operation may prevent the correct achievement of the next read operation. These malfunctions are modeled by two types of faults: Un-Restored Write Faults (URWFs) and Un-Restored Read Faults (URRFs). The formal definition of the functional fault modeling of URRF and URWF is given in the next sub-section.

3.3.1 Functional Fault Modeling of URRF and URWF

In this section, both fault models URRF and URWF have been introduced and detailed with experimental results. The functional fault model of URRF is composed of the following FP:

$$\mathrm{FP_{URRF}}: <ad1(\mathrm{x}r_{\mathrm{x}}),ad2(\bar{x}r_{\bar{x}})/ad2(\mathrm{x})/ad2(\mathrm{x})>$$

where

- *ad1* $\neq$ *ad2* and *ad1*, *ad2* belong to the same column;
- $x,\bar{x} \in \{0,1\}$ and $\mathrm{x} \neq \bar{x}$.

An *r*x is performed on a core-cell *ad1*. Then, an r$\bar{x}$ is performed in another core-cell *ad2* (*ad1* $\neq$ *ad2*) belonging to the same column than *ad1*. This read operation returns a faulty logic '$\bar{x}$' on the memory output and has inverted the core-cell content (from a logic 'x' to a logic '$\bar{x}$').

The functional fault model of URWF is composed of the following FP:

$$\mathrm{FP_{URWF}}: <ad1(\mathrm{y}w_{\mathrm{x}}),ad2(\bar{x}r_{\bar{x}})/ad2(\mathrm{x})/ad2(\mathrm{x})>$$

where

- *ad1* $\neq$ *ad2* and *ad1*, *ad2* belong to the same column;
- $\mathrm{y,x},\bar{x} \in \{0,1\}$ and $\mathrm{x} \neq \bar{x}$.

A *w*x is performed on a core-cell *ad1*. Then, an r$\bar{x}$ is performed in another core-cell *ad2* (*ad1* $\neq$ *ad2*) belonging to the same column than *ad1*. This read operation returns a faulty logic '$\bar{x}$' on the memory output and has inverted the core-cell content (from a logic 'x' to a logic '$\bar{x}$').

3.3.2 Experiments

For the study of the electrical phenomena behind URRFs and URWFs, it is practical to consider a generic delay fault produced on the pull-up of one of the two bit lines, e.g., bit line BLB. This simplification is allowed because the resistive-open defects Df1 and Df5 (the other defects do not lead to faulty behavior) cause URRFs and URWFs by producing a delay in the pull-up of one of the two bit lines (Dilillo et al. 2005d). Defect Df1 has its symmetric on the other side of the pre-charge circuit that leads to similar consequences, but requires inverted detection sequences.

As mentioned in the previous section, the requirement to detect a pre-charge faulty behavior is one read operation acted just after an incomplete pull-up of bit line BLB (BL in the symmetric case). The incorrect balancing of the two bit lines involves a premature ΔBL ($= \delta$) value that may disturb the core-cell content reading. In order to emphasize the delay in the pull-up of bit line BLB, it is useful that just before the read operation bit line BLB is pulled down as much as possible. This is possible by acting an *r*1 or *w*1 operation on a core-cell belonging to the same column before the *r*0 operation, as shown in Fig. 3.5. First, a *w*1 (or *r*1) is acted on core-cell $\mathrm{CC_{x,j}}$. At the end of this operation bit line $\mathrm{BL_j}$ is at VDD and

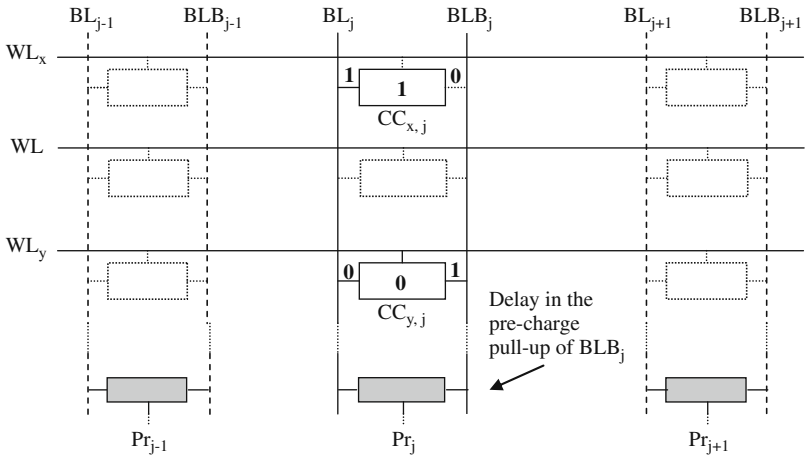


Fig. 3.5 Scheme for URWFs and URRFs sensitization

bit line BLB_j is at logic '0' (close to 0 V). During the pre-charge phase, the pull-up of bit line BLB is operated. When the following $r0$ operation is acted on core-cell $CC_{y,j}$, the pull-up of bit line BLB is not completed because of the faulty action of the pre-charge circuit. Consequently, at the beginning of this read operation the two bit lines are not equalized. In fact $V_{BL} = VDD$ and $V_{BLB} = VDD–\delta$. During the $r0$ operation, bit line BL is discharged by core-cell $CC_{y,j}$ left node, leading to

$$V_{BL} = VDD - \Delta BL \tag{3.1}$$

On the other side, bit line BLB is partially charged (with an increased voltage level of γ) by the right core-cell node (which is at VDD) reaching

$$V_{BLB} = VDD - \delta + \gamma \tag{3.2}$$

The read operation is acted by a sense amplifier that detects the differential voltage between the two bit lines. In the 130 nm memory considered in Dilillo et al. (2005d) and Dilillo et al. (2007), this difference has to be at least 80 mV for a correct read operation. In particular, for a correct $r0$ it is necessary that

$$V_{BL} - V_{BLB} = \Delta BL < -80\,mV (> +80\,mV \text{ for a } r1) \tag{3.3}$$

In the case of a faulty pre-charge circuit, using expressions (3.1) and (3.2), the result at the end of the $r0$ operation is the following one:

$$V_{BL} - V_{BLB} = (VDD - \Delta BL) - (VDD - \delta + \gamma) = \delta + \Delta BL - \gamma \tag{3.4}$$

When the value of δ is of the same order or larger than ΔBL, the condition for a correct read operation is not achieved, and in particular two different cases are possible:

Uncertain read: – 80 mV < $V_{BL} - V_{BLB}$ < 80 mV, the voltage difference is far from the safety value given in expression (3.3), and hence is in a range that does not allow the sense amplifier to act a safe read operation.

Incorrect read: $V_{BL}-V_{BLB} > 80$ mV, '0' is expected, but '1' is read.

As mentioned before, there are two types of fault that model this faulty behavior. The first one is the URRF, whose detection sequence is $r0$ ($r1$) operated on a core-cell and $r1$ ($r0$) on another core-cell connected to the same bit lines. The second one is the URWF, whose detection sequence is $w0$ ($w1$) operated on a core-cell and $r1$ ($r0$) on another core-cell connected to the same bit lines.

Dilillo et al. (2005d) and Dilillo et al. (2007) detail some experiments on URRF and URWF for the case study of a pre-charge circuit affected by defect Df1, whose waveforms are reported in Fig. 3.6. For the case (a), the detection pattern employed is the one of URRF and for case (b) it is the one of URWF. In both cases, the same resistance value has been chosen for defect Df1.

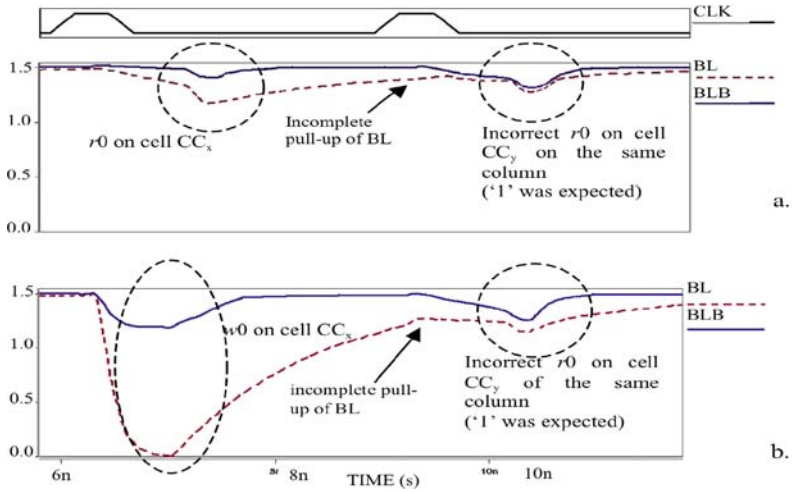


Fig. 3.6 Simulation with defect Df1 in the pre-charge circuit: URRF and URWF

Case a, URRF: A logic '0' is previously stored in core-cell CCx. An $r0$ is operated on core-cell CCx. At the end of this operation, the pre-charge circuit pulls up the bit lines connected to CCx, in particular the bit line BL that is at a lower voltage level than VDD. At the end of the pre-charge action, bit line BL is not at VDD as expected, because of the presence of defect Df1. Consequently, the following read operation on core-cell CCy, belonging to the same column and storing a logic '1,' is incorrect. In fact, $\Delta BL = V_{BL} - V_{BLB} < 0$ corresponds to an $r0$, while a logic '1' is expected.

Case b, URWF: A *w*0 is operated on core-cell CCx. At the end of this operation, the pre-charge circuit pulls up the bit lines connected to CCx, in particular BL that is at logic '0' (0 V) after the *w*0 operation. At the end of the pre-charge phase, bit line BL is not at VDD as expected, because of the presence of defect Df1. Consequently, the following read operation on core-cell CCy, belonging to the same column and containing a logic '1,' is incorrect. In fact, $\Delta BL = V_{BL} - V_{BLB} < 0$ corresponds to an *r*0, while a logic '1' is expected.

Figure 3.6 qualitatively shows that the sensitization pattern for the URWF is more effective than the one used for URRF, for the same defect (Df1) and same operational conditions. This analysis can be extended to a set of simulations, whose results are given in the graph of Fig. 3.7 (Dilillo et al. 2005d, 2007). For these simulations, the two sensitization sequences *r*0/*r*1 and *w*0/*r*1 are performed on a pre-charge circuit affected by defect Df1 (from 100 Ω to 10 kΩ) on bit line BL, with the following operating conditions:

- *Temperature:* 125°C
- *Process corner:* slow
- *Supply voltage:* 1.35 V
- *Df1 resistive value:* 100 Ω < RDf1 < 10 kΩ

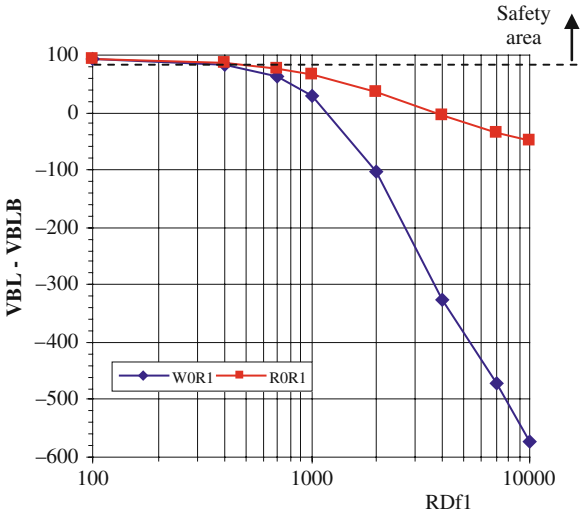


Fig. 3.7 Parametric simulations of the sequences *w*0*r*1 and *r*0*r*1, in presence of defect Df1

These conditions represent the worst case conditions that allow the easiest detection of the faults.

Figure 3.7 shows that for similar resistance values, the points relative to the *w*0*r*1 sequence correspond to $V_{BL} - V_{BLB}$ values that are less close to the *safety area* ($V_{BL} - V_{BLB} > 80$ mV, minimum differential voltage for a successful read operation)

than the equivalent points of the *r*0*r*1 sequence. Similar results can be obtained for other operational conditions and also for defect Df5.

The simulation results clearly show that the test procedure that sensitizes URWFs is always much more performing than the one that sensitizes URRFs. This means that the URRF sensitization pattern can be used for the test of pre-charge circuits, but is effective only for high-resistance values of the defect.

Since the faults that affect the pre-charge circuit are caused by the delay on the precB signal propagation (see Fig. 3.4) or by the delay in the pull-up and equalization of the bit lines, it is useful to investigate the link between clock cycle period and fault coverage. In this direction, Dilillo et al. (2005d) and Dilillo et al. (2007) present a study on the effectiveness of the URWF test pattern by varying clock cycle period. Figure 3.8 reports the results of this study, for the case of a pre-charge circuit affected by defect Df1 on bit line BL and with the following operating conditions:

- *Temperature*: 125°C
- *Process corner*: slow
- *Supply voltage*: 1.35 V
- *Df1 resistive value*: 100 Ω < RDf1 < 10 kΩ

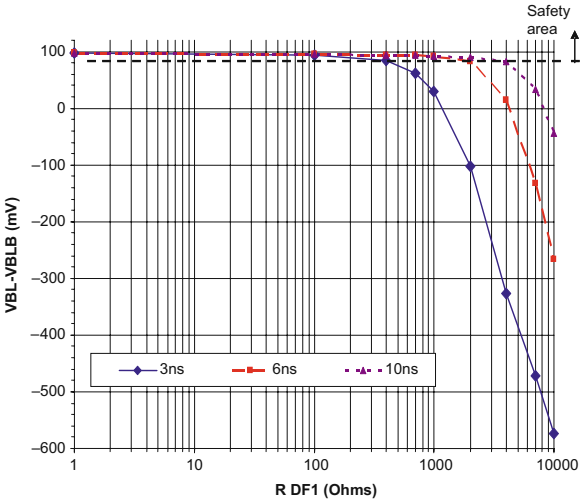


Fig. 3.8 Parametric simulations of the test sequence *w*0*r*1, in presence of defect Df1, at different frequency

Once again, these conditions represent the worst case conditions that allow the easiest detection of the faults.

Figure 3.8 shows that for similar resistance values of defect Df1, the points relative to higher frequency (smaller clock cycle period) correspond to $V_{BL} - V_{BLB}$ less close to the safety area ($V_{BL} - V_{BLB} > 80$ mV) than the equivalent points at lower frequency. Similar results are obtained for other operating conditions and also for a pre-charge circuit affected by defect Df5. These results show that in order to obtain

the best sensitization, it is useful to operate the test at highest operational frequency (*at-speed test*).

3.3.3 March Test Solutions Detecting URWFs

URWF detection requires writing certain data D in a core-cell and just after read the complementary value $\overline{D}$ in another core-cell of the same column. In the literature, there exist several March algorithms that embed the test pattern for URWFs detection and that are consequently able to cover the presence of resistive-open defects in the pre-charge circuits. Niggemeyer et al. (1998) proposes a particular implementation of the MATS+ test (complexity 5 N) depicted in Fig. 3.9.

Fig. 3.9 MATS+ algorithm

$$\Updownarrow (w0) \quad \Uparrow (r0, w1) \quad \Downarrow (r,1\ w0)$$
$$M_0 \qquad M_1 \qquad M_2$$

The first element M0 operates a $w0$ on all the core-cells of the memory. The second element M1 is implemented with the *fast_Y* addressing order, *column after column*. This means that, for a 1-bit transition of the column address decoder, all the possible transitions on the row address decoder are explored. This is allowed by the first degree of freedom of March tests (Section 1.3.4). Figure 3.10 shows the process of element M1 for three addresses, belonging to the same bit line couple.

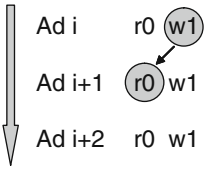


Fig. 3.10 Process of the element $\Uparrow(r0, w1)$

The sensitization phase is acted by the couple of operations $w1$ on a core-cell at address Ad_i and $r0$ on a core-cell at address Ad_{i+1}. The observation phase coincides with the $r0$ operation. This element is able to detect only a part of the possible URWFs and in particular those that are induced by the symmetric resistive-open defects and the asymmetric resistive-open defects placed on the right side of the pre-charge circuit (*c.f.* Fig. 3.4). For the test of the URWFs caused by the left-placed resistive-open defects (*c.f.* Fig. 3.4), it is necessary to operate element M2. The process of this element is similar to element M1 and the sensitization phase is acted by the couple of operations $w0$ on a core-cell at the address Ad_i and $r1$ on a core-cell at address Ad_{i+1}, as shown in Fig. 3.11.

MATS+ covers not only URWFs, but also a certain number of faults like SAFs. When URWFs are the only targeted faults, March Pre, proposed in Dilillo et al. (2006a), can be used. March Pre has the same effectiveness than MATS+ for URWF

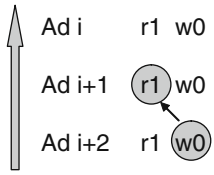


Fig. 3.11 Process of the element ⇓(*r*1, *w*0)

detection, but has a lower complexity, i.e., 2.5 N. The structure of this algorithm is depicted in Fig. 3.12.

Fig. 3.12 March Pre

$$\underset{M_0}{\Updownarrow(wO_v)}\ \underset{M_1}{\Uparrow(wE_{\bar{v}},\ rO_v)}\ \underset{M_2}{\Downarrow(wO_v,\ rE_{\bar{v}})}$$

Every memory structure is composed of an even number of core-cells, thus there is an even number of addresses. O_v denotes the data value $v \in \{0,1\}$ that can be written or read in the core-cells corresponding to odd addresses; Ev denotes the data value $v \in \{0,1\}$ that can be written or read in the core-cells corresponding to even addresses; and $O_{\bar{v}}$ is the complementary logic value of O_V, while $E_{\bar{v}}$ is the complementary logic value of E_v. For example, if v = '0' element M0 of March Pre, ⇕(wO_v), will act a *w*0 in all the core-cells of the memory array with odd addresses, with *any* addressing order (⇕). This means that element M0 has a complexity of 0.5 N (with N being the number of core-cells in the memory array). The process of elements M1 and M2 is more complex. For example, let us consider element M1 with v = 0. M1 acts a *w*1 on the core-cell with the first even address and, immediately after, an *r*0 on the core-cell with the first odd address. Next, a *w*1 is operated on the core-cell with the second even address, followed by an *r*0 operated on the core-cell with the second odd address, and so on. Both elements M1 and M2 have a complexity of 1N. Consequently, the total complexity of March Pre is 2.5N (= 0.5N + 1N + 1N).

The three elements of March Pre, performed on a hypothetical eight core-cells memory (= 8), with = 0 and one operation per clock cycle, are shown in Fig. 3.13.

Element M0: The first element has an upward addressing order. This element operates a *w*0 in the core-cells with odd addresses (<0 0 1>→**1**, <0 1 1>→**3**, <1 0 1>→**5**, and <1 1 1>→**7**).

Element M1: Element M1 operates alternatively a *w*1 and an *r*0 on the core-cells, with upward addressing order. The *w*1 operations are performed on the core-cells with even addresses (<0 0 0>→**0**, <0 1 0>→**2**, <1 0 0>→**4**, and <1 1 0>→**6**). The *r*0 operations are performed on the core-cells with odd addresses (<0 0 1>→**1**, <0 1 1>→**3**, <1 0 1>→**5**, and <1 1 1>→**7**), where the '0's have been written in the previous elements. In element M1, the two operations *w*1*r*0 induce the *sensitization*

	⇑ (wO_0)				⇑ ($wE_1; rO_0$)								⇓ ($wO_0; rE_1$)							
clock cycle / cell address	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
000					w1															r1
001	w0					r0													w0	
010							w1											r1		
011		w0						r0									w0			
100									w1							r1				
101			w0							r0					w0					
110											w1			r1						
111				w0								r0	w0							

Fig. 3.13 Process of March Pre in a 8-cell memory

of the URWFs, produced by the symmetric resistive-open defects and the asymmetric resistive-open defects placed on the right side of the pre-charge circuits. The *observation* of these faults is operated by the $r0$ operations.

Element M2: Element M2 operates alternatively a $w0$ and an $r1$ on the core-cells, with downward addressing order. In this element, the two operations $w0r1$ induce the *sensitization* of the URWFs, produced by the symmetric resistive-open defects and the asymmetric resistive-open defects placed on the left side of the pre-charge circuits. The *observation* of these faults is operated by the $r0$ operations.

As expected, the final number of operations performed by the algorithm is

$$\frac{1}{2}\mathrm{N} + \mathrm{N} + \mathrm{N} = 4 + 8 + 8 = 20\,(= 2.5\,\mathrm{N})$$

The March Pre algorithm can be used to detect the presence of resistive-open defects in the pre-charge circuits of any kind of memory. If the structure of the memory array is known, the complexity of March Pre can be considerably reduced. In order to produce an efficient test for the faults of all the pre-charge circuits, it is necessary to perform the two test patterns, $w0r1$ (operated as shown above), only one time per column. Further operations of the $w1r0$ patterns in the same column would be redundant.

Let us consider a memory composed of n_r rows and n_c columns. The number of core-cells of the array is the following:

$$\mathrm{N} = n\_r \cdot n\_c \tag{3.5}$$

If March Pre is applied to this memory, the complexity of the algorithm is the following:

$$\mathit{Complexity} = 2.5 \cdot \mathrm{N} = 2.5 \cdot n\_r \cdot n\_c\, \mathit{operations} \tag{3.6}$$

As mentioned above, for a correct test of a pre-charge circuit it is necessary to operate the URWF test pattern once for each column, thus acting on two core-cells for each column. Consequently, the set of operations that are sufficient for the test is the following:

- one $w0$ for the first core-cell of each column: n_c operations,
- one $w1$ for the second core-cell followed by an $r0$ of the first core-cell, for each column: $2{\cdot}n\_c$ operations,
- one $w0$ for the first core-cell followed by an $r1$ of the second core-cell, for each column: $2{\cdot}n\_c$ operations.

Thus, the complexity of this simplified version of the algorithm is the following:

$$Complexity = 5{\cdot}n\_c \text{ operations} \tag{3.7}$$

The ratio of the complexity of March Pre and its simplified version is given by

$$Complexity\ ratio = \frac{2.5 \cdot n\_r \cdot n\_c}{5 \cdot n\_c} = 0.5\, n\_r \tag{3.8}$$

For a 128 × 128 memory, the complexity of March Pre is performed with a number of operations given by

$$Complexity\,(\text{MarchPre}) = 2.5{\cdot}n\_r{\cdot}n\_c = 40960 \text{ operations}$$

while for the simplified version, the number of operation is

$$Complexity(\text{Simplifiedversion}) = 5{\cdot}n\_c = 640 \text{ operations}$$

On the market, there are memories with architectures more complex than the typical rectangular one. In case of memories with a very large number of core-cells, the rectangular structure is not exploitable as it would involve exceedingly long bit lines, with related problems of delay and capacitive coupling. In these cases, some architectural solutions propose the use of different bit line levels. For example, in a two-level architecture, the memory array is composed of a certain number of columns and one couple of primary bit lines with their primary pre-charge circuit corresponds to each column. The core-cells belonging to a column are not directly connected to the primary bit lines couple, but they are organized in smaller groups. For each group there is a secondary bit line couple with its own pre-charge circuit. All the secondary bit line couples are connected to the primary bit line couple through an access circuit driven by the address signals and the read write commands.

For memories organized with this architecture, the number of pre-charge circuits is larger than for the common rectangular structures, but the method used for the complexity reduction is always valid. The knowledge of the number of pre-charge

circuits and their locations is mandatory in order to produce a correct simplification of the test algorithm. In case the structure is unknown, no algorithm simplification can be operated and the process of the complete version of March Pre, Mats+, or other more complex algorithms is indispensable.

3.4 Conclusion

Resistive-open defects can affect the normal pull-up and equalization of the pre-charge circuit of SRAMs resulting in an unbalanced voltage level of the two bit lines that disturbs the read operation. The faults that model these malfunctions are the Un-restored Write Fault and the Un-Restored Read Fault. Experimental results show that not all the defect locations in the pre-charge circuit lead to an actual faulty behavior. Moreover, the URWF sensitization pattern is the more efficient than the URRF pattern. Other experimental analyses focusing on the frequency dependency demonstrate that at-speed test is necessary in order to obtain the best fault coverage. Several March tests are able to cover URWFs, such as MATS+ and March Pre. An efficient test reduction technique is applicable when the structure of the core-cell array is known.

Chapter 4
Resistive-Open Defects in Address Decoders

4.1 The SRAM Address Decoder

In memories, address decoders are used to access the core-cells where the data are physically stored. In fact, these decoders allow the selection of one core-cell (in bit-oriented memories) or a group of core-cells (in word-oriented memories) for the read/write operations. A 1 Mbit memory chip, externally organized as 1 M addresses of 1-bit words, would require one million word lines. The huge dimension of such address decoder would require a prohibitive silicon area. For this reason, two-dimensional addressing schemes are more suitable, with a row decoder for the word lines (WLs) and a column decoder for the bit lines (BLs). A simplified view of a two-dimensional addressing scheme is presented in Fig. 4.1. It represents a 4×4 decoder architecture composed of the two address decoders, one on the left and one on the top of Fig. 4.1. The memory array and the input/output circuitry are also depicted in this figure.

The size of each decoder, including the area required for the wires (bit/word lines), is proportional to $\sqrt{n}$, where n is the capacity of the memory expressed in number of bits. In the case of a 1 Mbit memory, the required decoder area is reduced from n to $2\sqrt{n}$, thus leading to the following reduction factor:

$$Reduction\,factor = \frac{n}{2\sqrt{n}} = \frac{1}{2}\sqrt{n} = \frac{1}{2} \cdot 1024 = 512 \tag{4.1}$$

Let us consider again the architecture shown in Fig. 4.1. For each row, the selection command is WLS_i that performs the selection of one word line. The selection command of each column is BLSi that allows the read/write access to each core-cell of the column.

In modern memories, the number of addresses is very large and consequently the address selection is done by row and column decoders through different levels of decoding. For this reason, memory elements like pre-decoders and post-decoders are very common. An implementation of a simple column decoder based on NOR gates is depicted in Fig. 4.2. This decoder is a 2–4 pre-decoder, which is currently used in industrial embedded memories. The structure has two inputs A0 and A1, synchronized with the BLEN (bit line enable) signal, and four outputs BLS0 to

A. Bosio et al., *Advanced Test Methods for SRAMs*,
DOI 10.1007/978-1-4419-0938-1_4, © Springer Science+Business Media, LLC 2010

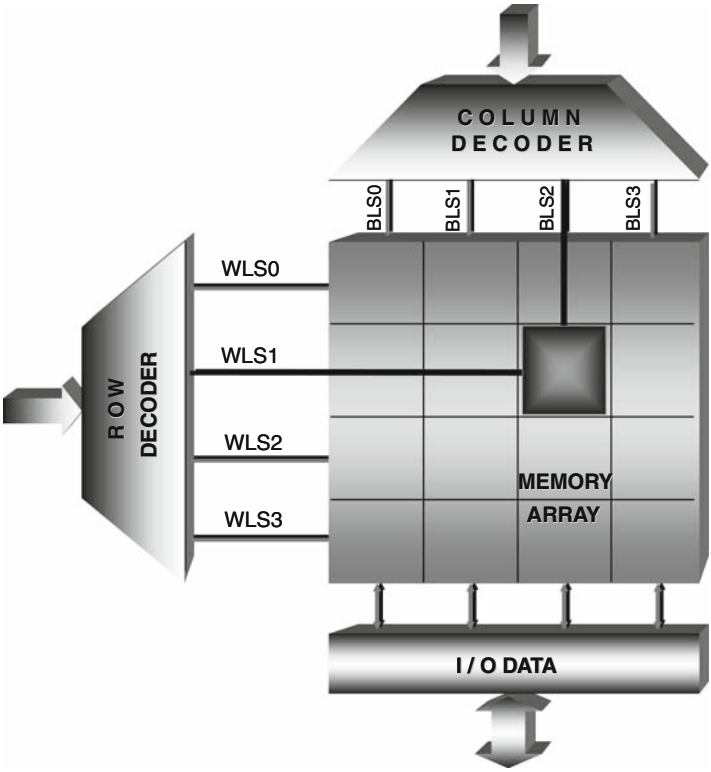


Fig. 4.1 A 4 × 4 decoder architecture

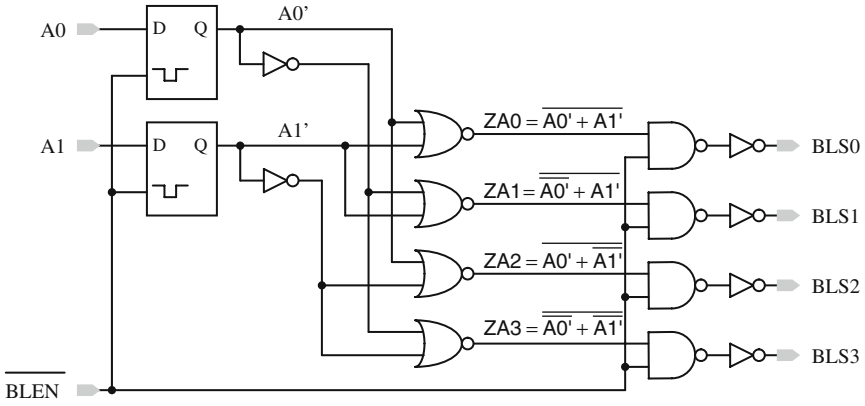


Fig. 4.2 A 2–4 NOR-based address Y-pre-decoder

BLS3. The decoding operations are made by the four NOR gates in the figure. The bit line selection signals are then re-synchronized with the BLEN signal before the next amplification stage.

Table 4.1 Truth table of the 2 × 4 Y-pre-decoder

A0	A1	BLS0	BLS1	BLS2	BLS3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

The truth table of this decoder is given in Table 4.1. The first column reports the four ($=2^2$) possible combinations of the two inputs; the remaining columns represent the output values of the four bit line selection signals.

It is useful to note that one and only one output signal is high for each input combination (A0A1). This means that any other different scenario (like all BLSi low or more than one BLSi high) indicates a faulty behavior of the memory.

4.2 Analysis of Resistive-Open Defects in the Address Decoder

In order to study the behavior of faults affecting the address decoder, Dilillo et al. (2003) proposes the injection of open and resistive-open defects in a 2-bit word line decoder, as depicted in Fig. 4.3. This decoder represents the homologous of the column pre-decoder shown in Fig. 4.2 and is also composed of NOR gates. The NAND gates and inverters are used for synchronization and buffering, respectively.

The defects affecting the address decoder can be *inter-gate* defects if they are located between gates or *intra-gate* defects if they are located inside the gate. Since only intra-gate-open and resistive-open defects can lead to dynamic faults, they are the only ones considered in this chapter for a detailed analysis. In particular, only the defects placed in the parallel plane of the decoding gates lead to dynamic faults. Considering the decoding NOR gate in Fig. 4.3, an open defect may be located in the drain, source, or gate nodes of transistors TN1 or TN2. For example, in Fig. 4.3, an open defect has been inserted in the drain node of transistor TN2. This defect causes an ADOF as shown in the next section.

4.3 Analysis and Test of ADOF

A pure open defect, like the one indicated in Fig. 4.3, disturbs the regular operation of the pull-down of the NOR-gate. The malfunction results in a missing de-selection of one word line. The consequence is that two word lines can be selected at the same time, so two memory core-cells (or words) are accessed during the same read/write operation (Dilillo et al. 2003).

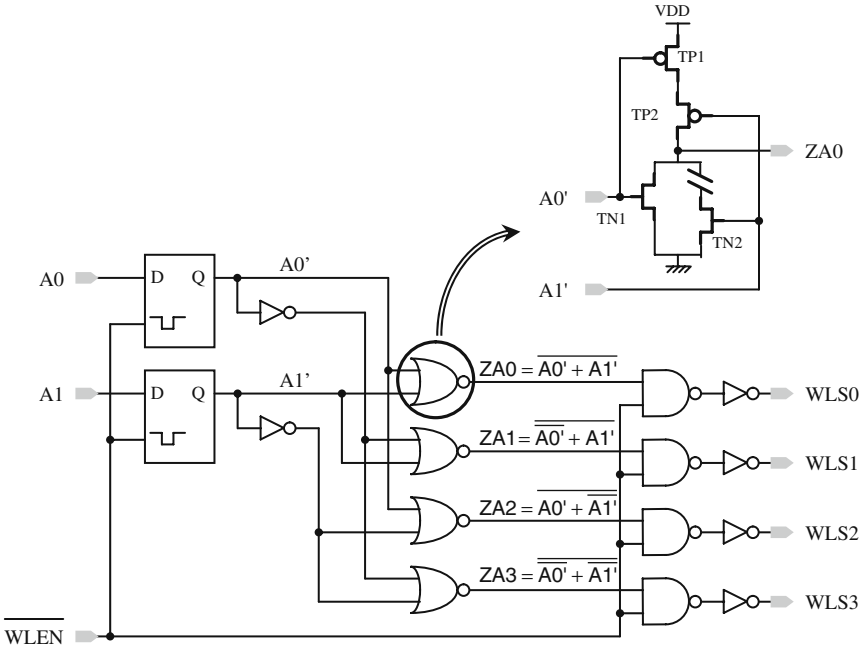


Fig. 4.3 A 2-bit NOR-based address decoder

4.3.1 Functional Fault Modeling of ADOF

A fault free 2-bit address decoder, as that shown in Fig. 4.3, is driven by signals A0 and A1 and activates only one word line at a time. For example, two consecutive accesses are done as follows:

a′. <A0 A1> = <0 0> ⇒ WLS0 is selected.
b′. Rising transition on A1: <A0 A1> = <0 1> ⇒ *WLS2 is selected and WLS0 is de-selected.*

Now, the sequential behavior of an ADOF is described by considering again the address decoder depicted in Fig. 4.3 with the open defect in transistor TN2. The corresponding waveforms are shown in Fig. 4.4 (Dilillo et al. 2003). As mentioned before, the presence of an ADOF induces a faulty selection of two word lines for the following two accesses:

a. <A0 A1> = <0 0> ⇒ WLS0 is selected.
b. Rising transition on A1: <A0 A1> = <0 1> ⇒ *WLS2 is selected and WLS0 remains selected.*

In this case, the open defect in transistor TN2 prevents the pull-down of node ZA0 that remains at logic ‘1’ due to the memory effect of this node.

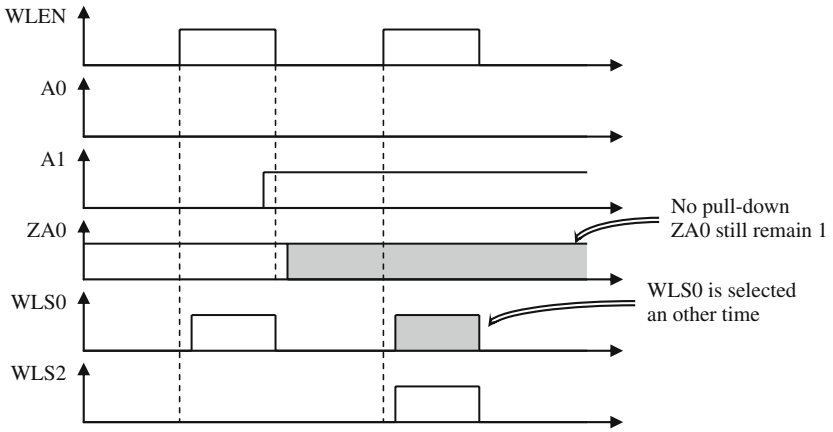


Fig. 4.4 NOR-based address pre-decoder waveforms in the presence of an ADOF

In this example, WLS0 (address <0 0>) is first selected and WLS2 (address <0 1>) is selected right after. Between the two addresses only one bit changes. The single switching of the address bits is a requirement for the fault sensitization because a two bit transition like <0 0> → <1 1> would activate both transistors TN1 and TN2 in the parallel plane of the NOR-gate. In this case, the discharge (pull-down) of node ZA0 would be correctly operated and WLS0 would be correctly de-selected, despite the presence of the open defect. This shows that a condition for ADOF detection is the use of an addressing sequence with a Hamming distance of 1 (Hd = 1), i.e., for each access the address must operate only one bit changed with respect to the previous one.

Considering the address decoder in Fig. 4.3, an effective detection sequence for ADOFs is the following one:

1′. Write ZERO at address <0 0> ⇒ *WLS0 is activated.*
2′. Write ONE at address <0 1> ⇒ *WLS2 is activated. WLS0 remains active because of the ADOF. A write '1' operation is abnormally performed in the two core-cells.*
3′. Read at address <0 0> ⇒ *ZERO is expected, ONE is read.*

Phases *1′* and *2′* are useful for sensitization and phase *3′* for observation. This diction pattern has been proposed in Sachdev (1996, 1997), where the author presented an algorithm (here referred to as Sachdev's algorithm) that produces a similar sequence which sensitizes and detects the ADOFs through the whole memory.

Considering the above statement, the functional models of ADOF is composed of the following two FPs:

– $\mathbf{FP1_{ADOF}}$: <*ad1*(xw_1), *ad2*(yw_0)/*ad1*(0)/*ad1*(–)>
– $\mathbf{FP2_{ADOF}}$: <*ad1*(xw_0), *ad2*(yw_1)/*ad1*(1)/*ad1*(–)>

where

- *ad1* $\neq$ *ad2* and Hd(*ad1*, *ad2*) = 1. Hd is the hamming distance between *ad1* and *ad2*;
- x,y $\in$ {0,1}.

4.3.2 Experiments

Figure 4.5 shows the first selection path of the address decoder presented in Fig. 4.3. On the left side, the electrical circuit of the NOR gate is depicted at transistor level. A resistive-open defect can be injected in transistor TN2 in three possible locations: on the source node, the drain node, or the gate node. Simulations of this sub-circuit of address decoder, by varying the resistance value of the defect in a range between 1 Ω and 2 MΩ, have been presented in Dilillo et al. (2003).

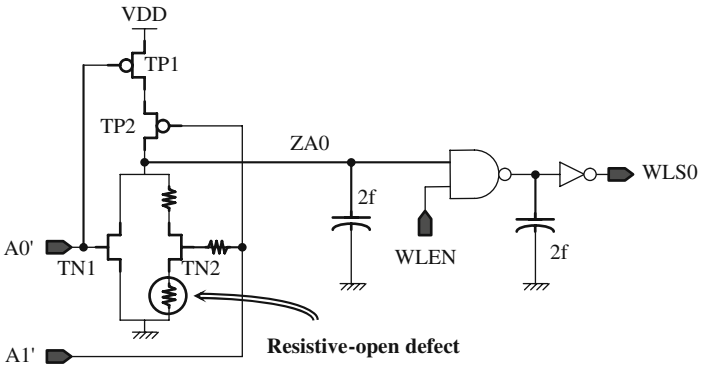


Fig. 4.5 Simulation scheme for intra-gate resistive-open defect injection

Figure 4.6 shows the waveforms of simulations performed for a resistive-open defect placed in the source node of transistor TN2. These waveforms are obtained with the following input transition sequence:

$$< \mathrm{A0\,A1} > = < 0\,0 > \rightarrow < \mathrm{A0\,A1} > = < 0\,1 >$$

The output of the NOR gate operates a transition from logic '1' to logic '0' (a pull-down is acted). This transition has a delay that increases linearly with the resistance value of the injected defect. This delay slows down the de-selection of WLS0.

The injection of resistive-open defects on the gate or drain nodes leads to similar delay effects. Figure 4.7 gives simulation results reported in Dilillo et al. (2003), where the three locations are explored. The delay/resistance ratio is always linear, but the circuit is less sensitive if the defect is placed on the gate node of the transistor.

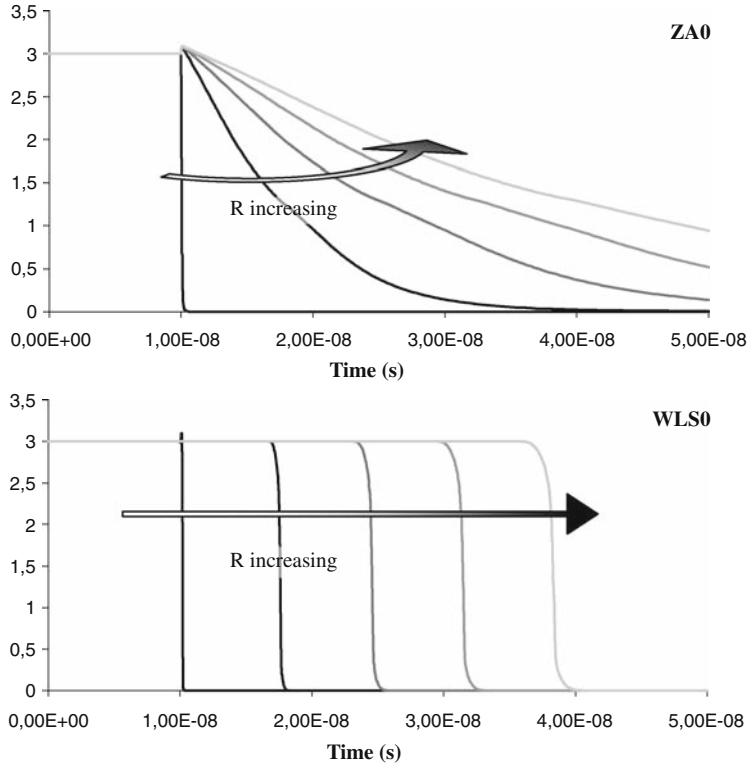


Fig. 4.6 Variable delay in WLS0 de-selection depending on the resistive-open defect value (defect inserted in the source of TN2)

This fact can be explained with the absence of bias current in CMOS transistors, which are consequently less sensitive to resistive-open defects in the gate nodes.

The consequences of a resistive-ADOF on the behavior of the address decoder are shown in Fig. 4.8. After the following input transition

$$< \mathrm{A0\,A1} > = < 0\,0 > \rightarrow < \mathrm{A0\,A1} > = < 0\,1 >$$

Word line WLS2 is correctly selected, while word line WLS0 remains selected erroneously for a certain time due to the delay in the pull-down of the defective NOR-gate. Considering the resistance value of the defect, three different situations can be distinguished:

a. In case of a *high resistive-open defect*, the circuit behaves as in the presence of a pure open, leading to an ADOF. This can be explained from Fig. 4.6. For high-resistive defects, the delay on de-selection of word line WLS0 is larger than the

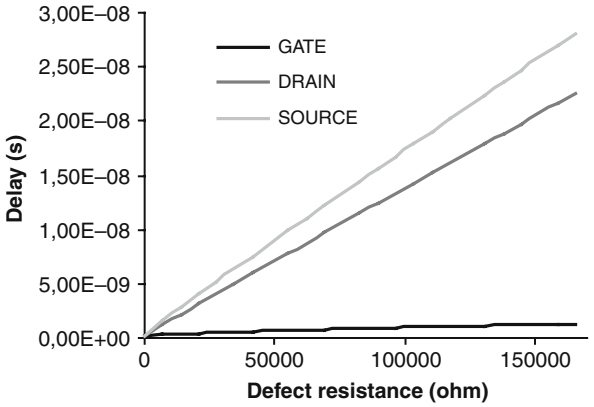


Fig. 4.7 Variable delay in WLS0 de-selection depending linearly on the resistive-open defect size (defect placed in the source, drain, and gate of transistor TN2)

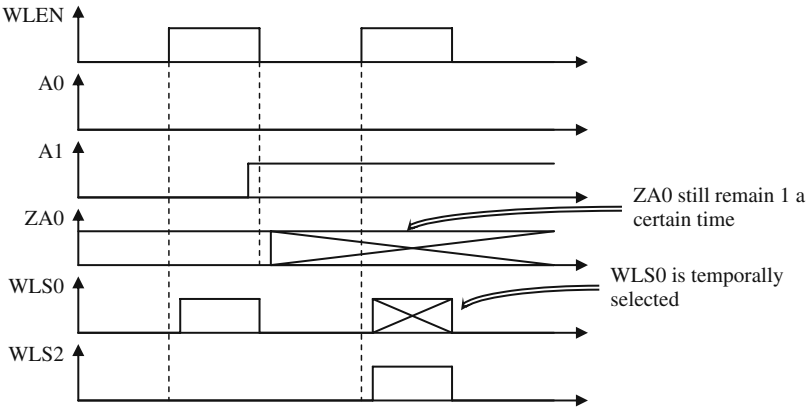


Fig. 4.8 NOR-based address pre-decoder waveforms with a resistive-open defect

clock period, thus two word lines are selected during the whole duration of the current read/write operation.

b. In case of a *very low resistive-open defect*, the delay perturbation is irrelevant. In other words, the delay in the de-selection of a word line lasts time which is too short to provoke a faulty double word line selection.
c. In case of *intermediate resistive-open defect*, the delay produced during the de-selection of word line WLS0 is not negligible. Word line WLS2 is correctly selected with a simultaneous selection of word line WLS0, during a certain time. This time, there is a high probability that a fault occurs, with effects similar to a full ADOF.

The Sachdev's algorithm, effective for ADOF detection, can also be effective for resistive-ADOFs if the defect induced delay leads to a double selection that exceeds the sensing phase of the write operation. Note that the Sachdev's algorithm has the sensitization phase during a write access.

4.3.2.1 Timing Constraints

The previous sub-section has shown that ADOFs and resistive-ADOFs can be treated with the same sensitization strategy. In particular, ADOFs are just a sub-class of resistive-ADOFs. For ADOFs and resistive-ADOFs, the sensitization requires the same conditions: for any open or resistive-open defect in the parallel plane of address decoder gates, all two-pattern sequences with Hamming distance $\mathrm{Hd} = 1$ have to be applied. In the presence of ADOFs or resistive-ADOFs, a double simultaneous access is produced only if this condition is satisfied (Sachdev 1996, 1997).

The timing constraints for the fault observation are different for ADOFs and resistive-ADOFs (Dilillo et al. 2006b). The waveforms in Fig. 4.9 refer to the address decoder with an injected resistive-open defect, as shown in Fig. 4.3. In these waveforms, T_s is the latch (the loop of two inverters in the core-cell) setup time, T is the sum between T_h (latch hold time) and T_{gate} (delay of address decoder logic gates), and δ is the delay due to a fixed resistive value of the defect. Three significant cases of timing address transition are considered.

In Fig. 4.9a, the input signal A1 switches to logic '1' while the synchronization signal WLEN is not active (as shown in Fig. 4.3, WLEN is active low). This involves that the falling transition of ZA0 is done after a delay $T + \delta$. In this case, there is no faulty behavior because there is enough time for a correct pull-down of node ZA0.

In the second case (Fig. 4.9b), the address transition occurs when WLEN is active. As for the previous case, there is enough time for the pull-down of node ZA0 and consequently no malfunction occurs.

Finally, in Fig. 4.9c, the address transition occurs exactly at a time T_s before the WLEN de-selection. In this case, WLS0 is erroneously selected for a certain time. This third case represents the best test condition to observe resistive-ADOFs. In fact, the smallest resistance value of the defect producing resistive-ADOFs is detectable *independently of the clock frequency*. If this condition (address transition close to WLEN deactivation) cannot be warranted during test, only at-speed test allows to reach the best compromise for a good sensitization of resistive-ADOFs.

Considering the previous remarks, two statements for ADOFs and resistive-ADOFs detection are given (Dilillo et al. 2006b):

1. In the presence of ADOFs, the faulty behavior of the circuit induces a double addressing for the whole sensing phase of operations. No particular timing constraints are required.
2. In the presence of *intermediate* resistive-ADOFs the faulty double word line selection persists for a part of the sensing phase of the operation. The occurrence of malfunctions is strictly related to the address transition. The best test condition

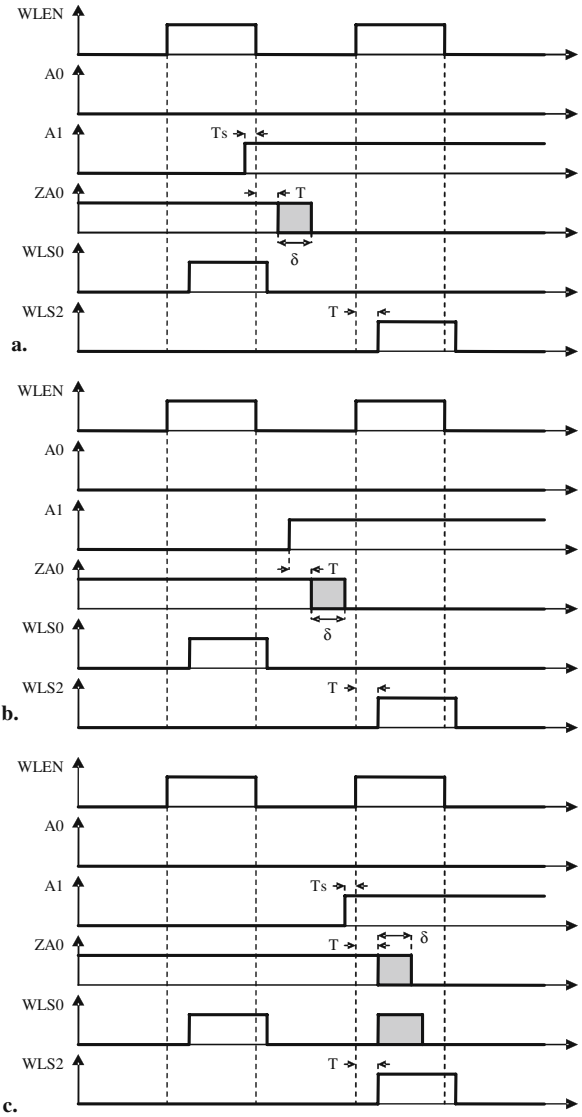


Fig. 4.9 Waveforms for a NOR-based address decoder with a fixed resistive-open defect and various timing address transition

is obtained when the address transition is close to WLEN de-selection. *At-speed test is the best compromise* in the absence of this condition.

Note that the two statements are valid not only for NOR-based address decoders but also for NAND-based address decoders. In both cases, the same requirements are needed for the fault sensitization. In the rest of the chapter, only ADOFs are mentioned, but all the analyses and test solutions are also valid for resistive-ADOFs.

4.3.3 March Test Solution Detecting ADOFs

In memory test, March test solutions are the most suitable because of their linear complexity and proven effectiveness in fault detection. In Klaus and Van de Goor (2001), two test conditions have been proposed in order to make the detection of ADOFs possible by using March tests:

1. The address sequence has to include all the pattern pairs with Hd = 1.
2. The considered March test must be already effective for address decoder static faults (AFs).

Condition 1 (Hd = 1) can be operated by exploiting the first of the six Degrees of Freedom of March tests (see Section 1.3.4) which is reported again below:

DOF I: Any arbitrary address sequence can be defined as an ⇑ sequence, as long as all addresses occur exactly once (⇓ is the reverse of ⇑). The fault detection properties are independent of the utilized address sequence (Nicolaidis 1985, Niggemeyer et al. 1998).

Condition 2 (AFs detection) is achieved by many March tests, and among these March C- (Marinescu 1982) is considered as a case study. March C- has six elements and a complexity of 10N and is presented in Fig. 4.10. Independently of the addressing sequence (DOF I), March C- is effective for detecting a large set of faults, but not ADOFs.

$$\{\Updownarrow(w0)\ \ \Uparrow(r0, w1)\ \Uparrow(r1, w0)\ \Downarrow(r0, w1)\ \ \Downarrow(r1, w0)\ \Updownarrow(r0)\ \}$$
M0 M1 M2 M3 M4 M5

Fig. 4.10 March C- algorithm

Now, the results of electrical simulations proposed in Dilillo et al. (2003) are given in order to verify the effectiveness of the two test conditions proposed in Klaus and Van de Goor (2001). These simulations refer to the NOR-based address decoder of Fig. 4.3 with a 130-nm technology. The main result of these simulations is that during the observation phase of the modified March algorithm some problems appear. The observation phase is done with the second element (M1) of March C-, when the ADOF involves a double word line selection, which happens during a read operation as shown in Fig. 4.11.

During the operation of March element M1, for the first address, Ad0 <A0 A1> = <0 0>, the two operations are properly performed: a logic '0' is read and a logic '1' is written. For the following address, Ad1 <A0 A1> = <0 1>, with only one bit transition (Hd = 1), an *r*0 is performed. In the presence of a defect causing an ADOF, the previously accessed core-cell (at address <0 0>) remains selected also during the following read operation. Consequently, two different logic values, a logic '1' stored at Ad0 and a logic '0' stored at Ad1, are simultaneously read on the same bit line couple (BL/BLB). This involves a possible fail of the

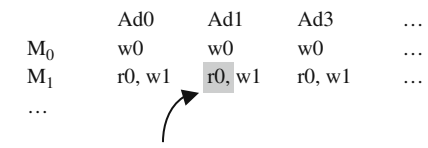


Fig. 4.11 ADOFs detection with March C- (address sequence with Hd = 1)

read access, but it does not ensure that the fault is observed, i.e., the reading of an unexpected value (revealing the ADOF) is not warranted.

In Dilillo et al. (2003), the March sequence proposed in Klaus and Van de Goor (2001) (see Fig. 4.11) has been applied to a 130-nm synchronous single-port SRAM with defective row decoder, with a resistive-open defect (R) placed on the source node of one NOR-gate, as in Fig. 4.3. The simulation results are reported below. As mentioned with more details in other chapters, the bit lines BL and BLB are the core-cell input/outputs lines, as shown in Fig. 4.12. Note that for the considered memory, the read operation has a detection limit of –/+ 80 mV for ΔBL ($\Delta BL = V_{BL} - V_{BLB}$). This is the minimal voltage difference between the two bit lines of the read core-cell, allowing a correct read operation. For example, for a $\Delta BL = 50$ mV the Sense Amplifier may return an uncertain (random) value.

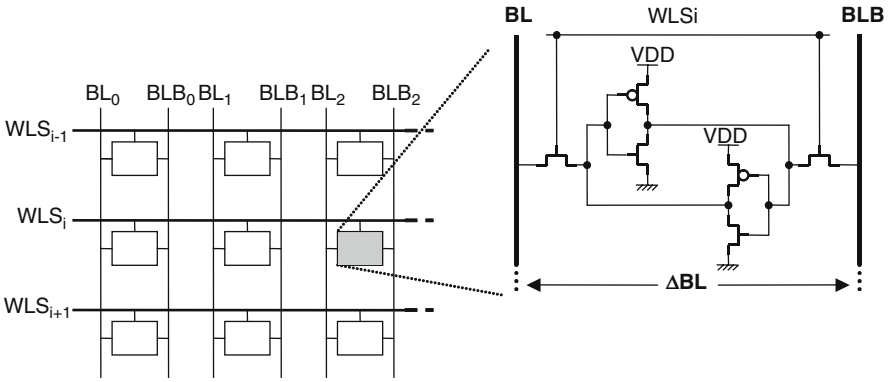


Fig. 4.12 Scheme of a 6-transistor SRAM core-cell with bit lines

The graphics in Fig. 4.13 present the results of these simulations. Figure 4.13a, b shows the simulation waveforms relative to a Sachdev-like (in grey) and a March-like (in black) test sequence, when the observation is done respectively with an *r*0 and an *r*1 operation. Let us consider the case with the observation phase operated by an *r*0 operation (waveforms on the left). The *r*0 operation is performed respectively during *phase c* of Sachdev's algorithm (see Section 4.3.1) and during element M1 of March C- (see Fig. 4.11). Obviously, both tests have been performed with an Hd = 1 addressing sequence in order to produce a double faulty word line selection due to the open defect in the row decoder. Figure 4.13a shows that Sachdev's algorithm,

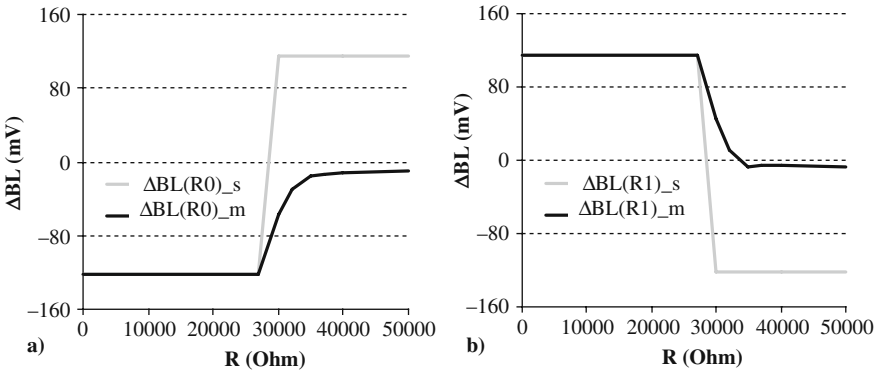


Fig. 4.13 Sachdev's algorithm vs. March C-

marked with ΔBL(R0)_s, is effective for ADOFs detection because a logic '1' is read instead of the expected logic '0,' for a resistance value of the defect $R \geq 27\ \text{k}\Omega$. On the other hand, the March C-, marked with ΔBL(R0)_m, does not ensure the fault detection because the read value is undefined for the same resistive value: $\Delta\text{BL} < |80|$ mV. In Fig. 4.13b similar results are shown for an $r1$ operation.

In conclusion, the Sachdev's algorithm is effective for the sensitization and the observation of ADOFs because the double addressing occurs during a write operation. Conversely, March C- is effective for fault sensitization (double word line selection during write operation), but the observation phase is not effective because the read of an unexpected value is not guaranteed. In fact, the double core-cell access occurs also during the read operation, thus causing two opposite values to be read on the same bit lines resulting in an uncertain read value.

This demonstrates that for a proper detection of ADOF, it is necessary to sensitize the fault during the write operation and observe it during a read operation.

4.3.3.1 Sachdev's Algorithm Converted in March Elements

Despite the fact that Sachdev's algorithm is effective to test ADOFs, an efficient March test solution still remains the most attractive solution, because this type of test has a very low complexity and a good effectiveness to cover a large number of faults. For this reason, in Dilillo et al. (2006b) the authors propose to convert the Sachdev's algorithm into March elements, considerably reducing its complexity. In order to attend this target, some Degrees of Freedom of March tests were exploited (see Section 1.3.4).

The three phases of Sachdev's algorithm can be graphically illustrated as in Fig. 4.14, where, between Ad0 and Ad1, Hd = 1, d = data ($d \in \{0, 1\}$), and $\overline{d}$ is its complementary logic value.

This pattern can be implemented in March elements. For this purpose, the Sachdev's pattern has to be reiterated for all addressable core-cells, as shown in

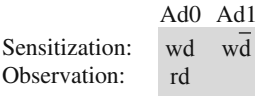


Fig. 4.14 Sachdev's pattern

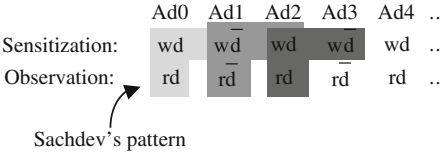


Fig. 4.15 Sachdev's algorithm adaptation

Fig. 4.15. Instead of performing sensitization and observation phases for each core-cell, a serial sensitization phase can be executed for all the core-cells of the core-cell array, by a serial write operation with an alternating data d and $\overline{d}$, followed by a global observation phase by a serial reading of all the core-cells.

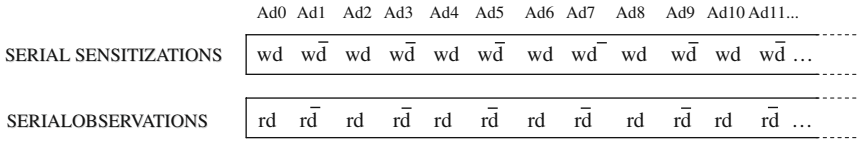


Fig. 4.16 Sachdev's algorithm adaptation into March elements

In Fig. 4.16, another graphical representation of the Sachdev's algorithm adaptation is given (Dilillo et al. 2006b).

This second representation can be easily interpreted as two March elements that are summarized in Fig. 4.17, where A is a logic value which starts with a logic '0' or a logic '1' and takes the complementary logic value for each new address; the addressing sequence has Hd = 1.

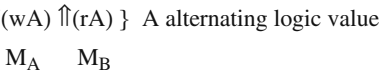


Fig. 4.17 New March elements for ADOFs detection

The possibility to change the data values during the execution of a March element is allowed by the fourth degree of freedom of March tests:

DOF IV: The data within a read/write operation does not necessarily have to be equivalent for all memory addresses as long as the detection probabilities of basic faults are not affected (Niggemeyer et al. 1998).

During the test operation of the March elements of Fig. 4.17, the electrical behavior of the memory is similar to the operation of Sachdev's algorithm (see Fig. 4.13). The double addressing due to the ADOF occurs during the write operation (sensitization phase). The observation phase is performed during the read operation without

uncertainty because there is not double core-cell (word line) selection with opposite data as for the solution proposed in Klaus and Van de Goor (2001).

In order to ensure that the proposed March elements cover all the ADOFs and resistive-ADOFs, it is necessary that the sequence of 2^m produced addresses (where m is the total number of address bits) contains all the $n \times 2^n$ single-bit transitions (where n is the bit-width of the considered decoder and 2 in the case of the decoder in Fig. 4.3) (Otterstedt et al. 1998). In other words, the necessary condition for complete detection is:

$$2^m \geq n \times 2^n + 1 \qquad 4.2$$

If this condition is not satisfied, it is necessary to add other two similar March elements with the reverse address sequence that imply the occurrence of the opposite single-bit transitions. The condition in Eq. 4.2 is often satisfied in industrial architectures.

The test complexity of the March elements is 2N (i.e., 2×2^n) for an n-cell memory and is low if compared to $(2n+1) \times$ N (i.e., $(2n + 1) \times 2^n$) of the Sachdev's algorithm. Moreover, for a BIST implementation, the Sachdev's algorithm requires a dedicated address generator, which cannot be reused for the operation of March tests. Conversely, the March-like solution is more attractive because the same address generator built for the operation of the two March elements can be also reused for additional March tests. A concrete BIST realization of an Hd = 1 address generator can be implemented with an LFSR (linear feedback shift register).

4.3.3.2 March iC-

The two March elements presented above allow complete sensitization and observation of ADOFs. In Dilillo et al. (2004a), the properties of this March elements have been embedded in existing March algorithms. One example of March algorithm improvement to cover ADOFs is the *March iC-*, which is essentially generated by modifying March C- by introducing a particular addressing order and a particular read/write data background sequence. These modifications do not change the original complexity of March C- and its capability to detect the former target faults.

The modifications to be introduced in March C- (shown in Fig. 4.10) are the following:

- Address sequence with Hd = 1.
- Alternating data value A_v, for the read or write operations, where v is the initial value ($\overline{v}$ indicates the complementary logic value of v).

With these modifications, the improved March C- becomes March iC-, which has the structure shown in Fig. 4.18 (Dilillo et al. 2004a).

In order to clarify the structure and the use of the March iC-, some elements of memory design and notations are introduced here. Every memory structure is composed of an even number of core-cells, thus there is an even number of addresses. Av denotes alternating data with starting value $v \in \{0,1\}$; $A_{\overline{v}}$ is its complementary

$$\{ \Uparrow (wA_v) \;\; \Uparrow (rA_v, wA_{\overline{v}}) \;\; \Uparrow (rA_{\overline{v}}, wA_v) \;\; \Downarrow (rA_{\overline{v}}, wA_v) \;\; \Downarrow (rA_v, wA_{\overline{v}}) \;\; \Updownarrow (rA_{\overline{v}}) \;\}$$

M0 M1 M2 M3 M4 M5

Fig. 4.18 March iC-, general version

value. If $v =$ '0' as starting value for a write operation (wA_0), the value written in the last core-cell will be a logic '1,' because of the even number of core-cells. If, after the wA_0 operation, a read operation with inverse addressing order is acted, the first value that is expected is $v =$ '1.' For this reason, in element M2 $\Uparrow$ ($rA_{\overline{v}}$,wA_v), the starting value for the alternative data to write is v and the starting read value in M3 $\Downarrow$ ($rA_{\overline{v}}$,wA_v) is $\overline{v}$. This is graphically shown in Fig. 4.19 where M2 and M3 are applied on a hypothetical 4-cell memory

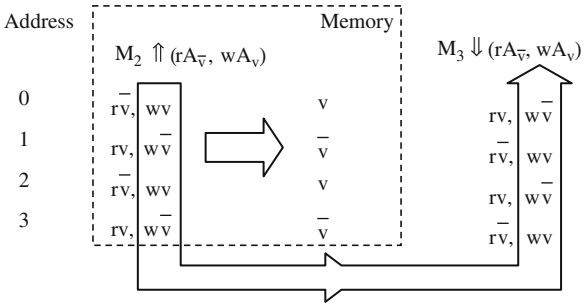


Fig. 4.19 Elements M_2 and M_3 applied in a 4-cell memory

The stored data after the application of element M2 are shown in the column in the middle of Fig. 4.19. The following read operation, made by element M3, starts with the expected data $\overline{v}$ at the address <1 0>, i.e., the last core-cell where M2 is executed.

4.4 Conclusion

When affected by intra-gate open and resistive-open defects, the address decoder may present dynamic faults such as ADOFs (Address Decoder Open Faults) and resistive-ADOFs. ADOFs are a sub-class of resistive-ADOFs and similar test conditions are valid for both of them. Electrical simulations demonstrate that ADOFs can be detected only when the sensitization phase involves a double addressing during the write operation. Some effective test solutions, based on Sachdev's pattern, allow to cover ADOFs. In particular, a compact test solution consisting of two March elements is particularly effective and can be embedded in other existing March tests without reducing the coverage of the original target faults. A case study of the March iC-, which is based on the well-known March C-, has been presented at the end of this chapter.

Chapter 5
Resistive-Open Defects in Write Drivers

5.1 The SRAM Write Driver

This section describes the structure of the write driver involved in the I/O circuitry of an SRAM. The first sub-section provides a global view of the I/O circuitry and the second sub-section details the write driver functioning.

5.1.1 Write Driver Within the I/O Circuitry

The I/O circuitry controls and observes BLs and BLBs during the write and read operations. A simplified view of an SRAM including the I/O circuitry is depicted in Fig. 5.1. The I/O circuitry is shared by BL couples selected by sub-Muxes which are themselves controlled by a SEL_i signal. The functioning of the sub-Muxes is as follows:

- If SEL_0=1 (SEL_1=. . .=SEL_m=0) then
  - BL_0=WD=SA
  - BLB_0=WDB=SAB
- If SEL_m=1 (SEL_0=. . .=SEL_{m-1}=0) then
  - BL_m=WD=SA
  - BLB_m=WDB=SAB

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

Before every read or write operation, BLs and BLBs are pre-charged at VDD. The write driver nodes (WD and WDB) and the sense amplifier nodes (SA and SAB) are also pre-charged at VDD by their own pre-charge circuitry, labeled as PRE in Fig. 5.1. During a read operation, the sense amplifier translates the weak

A. Bosio et al., *Advanced Test Methods for SRAMs*,
DOI 10.1007/978-1-4419-0938-1_5, © Springer Science+Business Media, LLC 2010

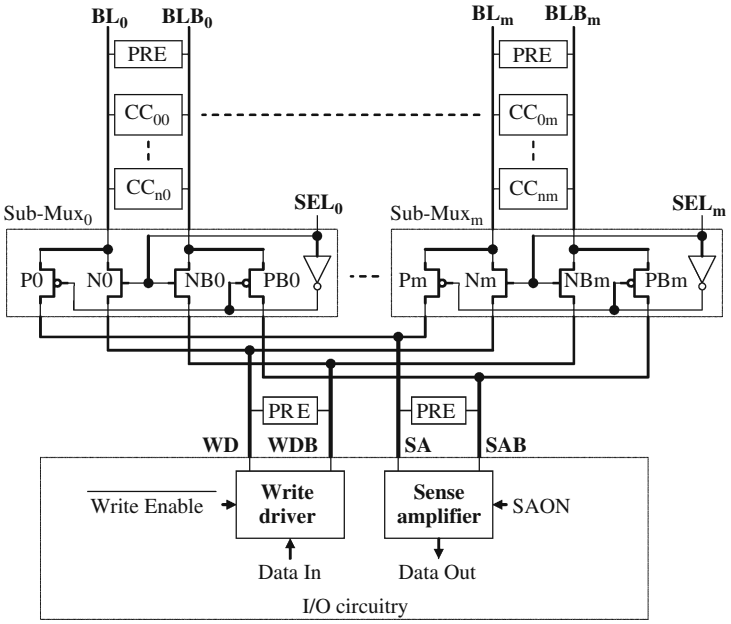


Fig. 5.1 The SRAM I/O circuitry

differential voltage between BL and BLB in a full-swing differential signal which is then interpreted as a digital signal to provide the logic data output. The functioning of the write operation is described in the next sub-section.

5.1.2 Operation Mode of the Write Driver

To perform a write operation, as the two bit lines are pre-charged at VDD before every operation, the write driver has just to act the pull-down of one of the two bit lines:

- BL for a *w*0 operation
- BLB for a *w*1 operation

A basic write driver structure is shown in Fig. 5.2. It is composed of a write control part and a driver part. The write control part receives the data that has to be written (DataIn) and the Write Enable signal (active at low level). This part controls the write operation with its two outputs, named AW0 and AW1. If DataIn = 0 and Write Enable is active, then AW0 =1 and AW1= 0. In that case, which corresponds to a *w*0 operation, the transistor Mtn1 acts the pull-down of the selected BL. Similarly, if DataIn =1, AW0 = 0 and AW1=1, which corresponds to a *w*1 operation, the transistor Mtn2 acts the pull-down of the corresponding BLB.

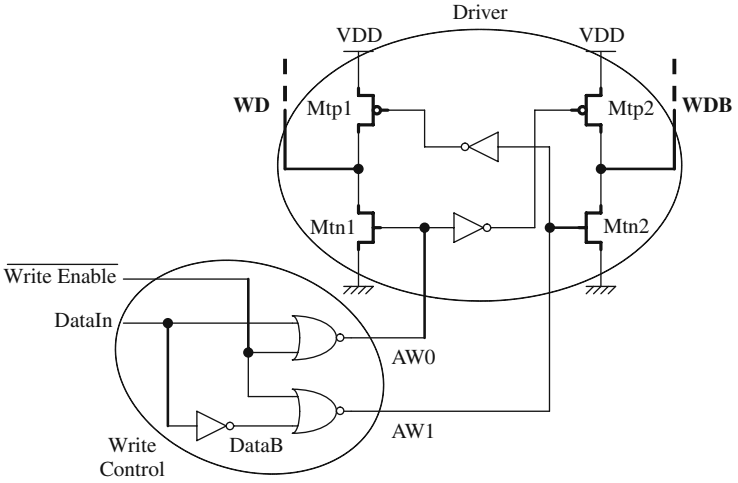


Fig. 5.2 Write driver structure

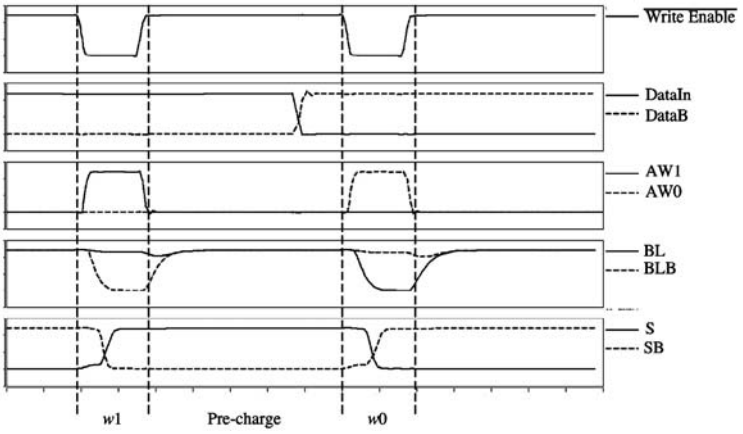


Fig. 5.3 Write driver functioning ($w1$ and $w0$ operations)

At this point, it is important to notice that usually signals AW0 and AW1 can never be set to a logic '1' at the same time.

The waveforms presented in Fig. 5.3 show how the write driver operates during two consecutive write operations. The write operations performed here are a $w1$ followed by a $w0$ in a core-cell that initially contains a logic '0.' Nodes S and SB are the state values of the core-cell.

5.2 Analysis of Resistive-Open Defects in the Write Driver

In this section, the resistive-open defects considered in the write driver are first described. Next, the impact of each defect is qualitatively analyzed and the corresponding fault modeling is presented. Finally, the fault modeling is validated with electrical simulations.

5.2.1 Defect Location

As shown in Fig. 5.4, nine resistive-open defect locations (Df1–Df9) have to be considered in the write driver (Ney et al. 2007a). The remaining defect locations can actually be considered as equivalent to one of these nine locations owing to the symmetry of the write driver structure. Consequently, only the left part of the write driver with defects Df1–Df4 has to be considered. Two defects (Df5 and Df6) are also placed in the inverter of the write driver control part and three defects (Df7–Df9) in one of the NOR gates of this part. Symmetric defects can be placed on the other NOR gate of the write control part and in the right part of the driver.

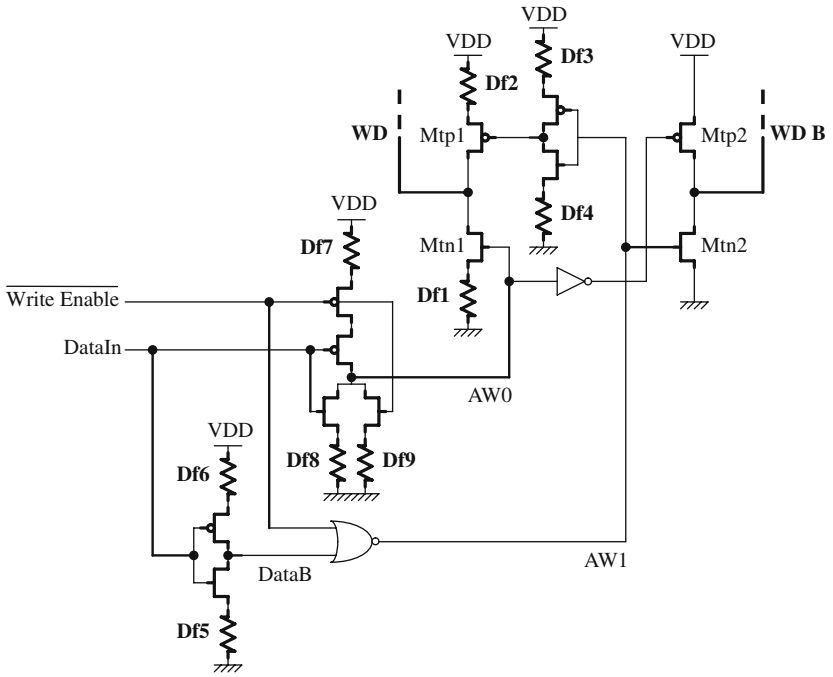


Fig. 5.4 Resistive-open defect considered in the write driver

5.2.2 Defect Incidence Analysis

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

- *Defect Df1:* This defect produces a delay in the discharge phase of BL during the write operation. The faulty behavior related to Df1 can be modeled by a Transition Fault (TF). This fault is a static fault and many classical March tests are able to detect it.
- *Defect Df2:* This defect induces a delay in the charging operation of BL during the write operation. 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 anyhow. Consequently, no faulty behavior occurs in this case.
- *Defect Df3:* This defect prevents to turn OFF transistor Mtp1. Consequently, Mtp1 is still turned ON. The worst case would occur when 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 five times stronger than the P plan (Mtp1 and Mtp2), thus 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. Thus, even if Mtn1 has to fight against Mtp2, the resulting level on BL is a logic '0.' Hence Df3 has no impact on the write driver functioning.
- *Defect Df4:* This defect produces an effect similar to Df2.
- *Defects Df5 and Df6:* During a write operation, one of the two bit lines is driven to logic '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 a Slow Write Driver Fault (SWDF).
- *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, Df8 has no impact on the write driver functioning.
- *Defect Df9:* A $w0$ operation can be performed by the write driver, i.e., AW0 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. However, Df9 prevents this pull-down and thus AW0 remains at a logic '1' a certain time depending on the size of defect Df9. Hence, the write driver continues to perform a $w0$ operation even if a read operation has to be done. This faulty behavior can be modeled as an Un-Restored Destructive Write Fault (URDWF) or an Un-Restored Write Fault (URWF).

Table 5.1 presents a summary of the fault models identified for each injected resistive defect.

Definitions of fault models reported in Table 5.1 are the following:

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.

Table 5.1 Fault models identified for each injected defect

Defect	Fault model
Df1	TF
Df2	–
Df3	–
Df4	–
Df5	SWDF
Df6	SWDF
Df7	TF
Df8	–
Df9	URWF URDWF

Slow Write Driver Fault (SWDF) (Van de Goor et al. 2004): 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 behavior results in the core-cell that does not change its data content.

Un-Restored Write Fault (URWF) (Adams and Cooley 1997): 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 core-cell belonging to the same I/O circuitry is not correctly acted.

Un-Restored Destructive Write Fault (URDWF) (Ney et al. 2007): The same definition as URWF but in addition to the faulty read operation, the core-cell content swaps.

Resistive-open defects in the write driver may be the consequence of a static fault (TF) as well as dynamic faults (SWDF, URWF/URDWF). The static fault is well known and is easily detected by classical March tests. Dynamic behaviors, represented by SWDF and URWF/URDWF, are analyzed in more detail in the next sections.

5.2.3 Simulation Set-Up and Results

In Ney et al. (2007a), the authors have examined the whole range of operating conditions with the aim of determining those conditions that maximize the fault detection probability. Simulations have been performed using a 65-nm technology by applying a number of different test patterns (according to the expected fault models defined above) with a defect size that ranges from a few ohms up to several megaohms and for different PVT parameters:

- *Process corner*: slow, typical, fast, fast n/slow p, slow n/fast p
- *Voltage supply*: 1.08 V, 1.2 V, 1.32 V
- *Temperature*: –30°C, 27°C, 110°C

Table 5.2 presents the conditions for maximum fault detection (i.e., the PVT conditions), the minimum detected resistance value, and the observed fault model.

Table 5.2 Summary of worst-case PVT corners for the defects in Fig. 5.4 and their corresponding minimum detected resistance and fault model

Defect	Process corner	Voltage (V)	Temp (°C)	Min Res. (kΩ)	Observed fault model
Df1	fast	1.08	–30°C	0.4	TF
Df2	–	–	–	–	–
Df3	–	–	–	–	–
Df4	–	–	–	–	–
Df5	fast	1.08	–30°C	128	SWDF
Df6	slow n/fast p	1.32	110°C	170	SWDF
Df7	fast	1.32	–30°C	9.5	TF
Df8	–	–	–	–	–
Df9	slow	1.08	–30°C	72 110	URWF URDWF

The first column (Defect) indicates the defect location in the write driver according to Fig. 5.4. The following three columns correspond to the electrical parameters which maximize the fault detection. The fifth column provides the minimum defect size that induces a faulty behavior. Finally, the last column gives the observed fault models.

5.3 Analysis and Test of SWDF

As shown previously, a SWDF can be produced by two resistive-open defects: Df5 and Df6 in Fig. 5.4. In this section, a Functional Fault Modeling (FFM) of these defects is first presented. Next, the conditions required to detect all SWDFs are detailed.

5.3.1 Functional Fault Modeling of SWDF

During a fault-free write operation, one of the two bit lines is driven to a logic '0' and the other one remains at VDD. However, in the presence of Df5 or Df6, this operation cannot be performed, especially when there are two successive write operations with opposite values. On this basis, SWDFs can be defined with four Fault Primitives (FPs) divided into two groups according to the defect induced (Df5 or Df6).

The first group corresponds to defect Df5:

- FP1: $<1w_0w_1/0/->$ 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.' The $w1$ operation is not performed.
- FP2: $<0w_0w_1/0/->$ A logic '0' 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.' The $w1$ operation is not performed.

The second group corresponds to defect Df6:

- FP3: $<0w_1w_0/1/->$ A logic '0' is initially stored in the core-cell. Then, a $w1$ is acted immediately followed by a $w0$. The core-cell remains at a logic '1.' The $w0$ operation is not performed.
- FP4: $<1w_1w_0/1/->$ A logic '1' is initially stored in the core-cell. Then, a $w1$ is acted immediately followed by a $w0$. The core-cell remains at a logic '1.' The $w0$ operation is not performed.

FP1 or FP2 (respectively FP3 or FP4) can be equally considered to model the effect of Df5 (respectively Df6).

5.3.2 Experiments

Here after, electrical simulations of the write driver in presence of Df5 and Df6 are detailed respectively. These simulations have been obtained using a 65-nm SRAM technology.

5.3.2.1 SWDF Due to Df5

Waveforms in Fig. 5.5 present the faulty behavior of the memory in presence of Df5 with typical PVT conditions (Process Typical, Voltage 1.2 V and Temperature 27°C) and a defect size of about 900 kΩ.

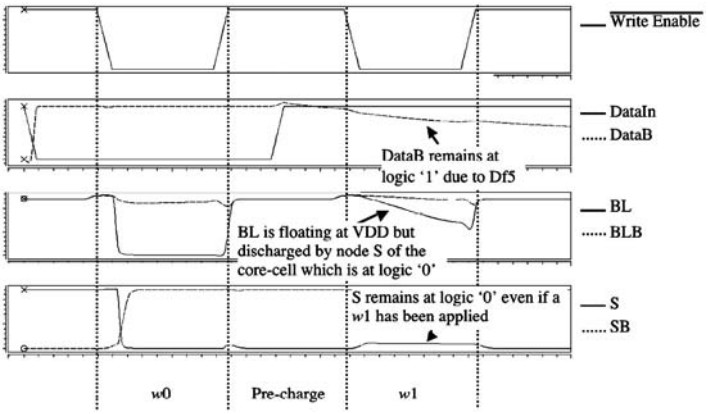


Fig. 5.5 Waveforms of FP1 simulation (Df5)

The simulation starts on a core-cell that initially contains a logic '1.' A $w0$ operation is first applied. Node DataIn is set to a logic '0' and node *DataB* is set to a logic '1' before the write operation. This first write operation is correctly acted on the core-cell which switches from a logic '1' to a logic '0.' Then, a $w1$ operation is

performed. Just before this operation, DataIn is set to a logic '1' but DataB remains also to a logic '1.' In that case, the pull-down of DataB cannot be performed due to the presence of Df5. The two nodes AW0 and AW1 are set to a logic '0.' Any write operation cannot be performed as the four transistors of the write driver (Mtp1, Mtn1, Mtp2, and Mtn2) are turned OFF. The two bit lines are floating at the VDD level.

5.3.2.2 SWDF Due to Df6

Waveforms in Fig. 5.6 present the faulty behavior of the memory in presence of Df6 with the same operating conditions as those used for Df5.

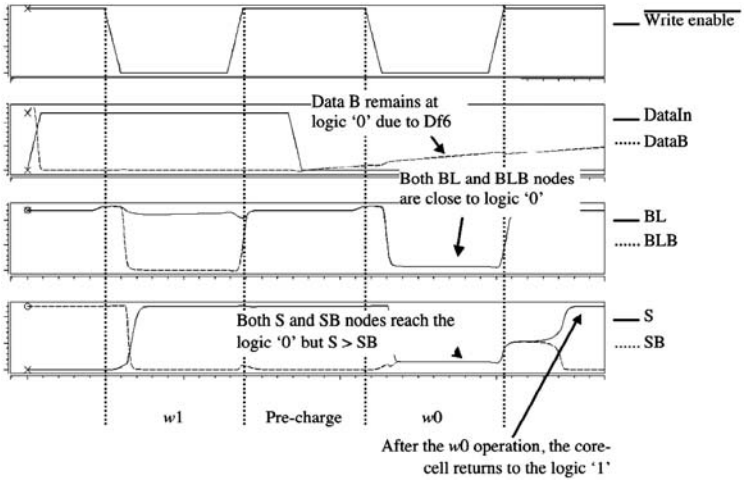


Fig. 5.6 Waveforms of FP3 simulation (Df6)

The simulation starts on a core-cell that initially contains a logic '0.' A $w1$ operation is first applied. Node DataIn is set to a logic '1' and node DataB is set to a logic '0' before the write operation. This first write operation is correctly acted and the core-cell switches from a logic '0' to a logic '1.' Then, a $w0$ operation is performed. Just before this operation, DataIn is set to a logic '0' but DataB also remains to a logic '0.' In that case, the pull-up of DataB cannot be performed due to the presence of Df6. The two nodes AW0 and AW1 are set to a logic '1.' This configuration is problematic as it means that the driver has to perform simultaneously a $w0$ (AW0=1) and a $w1$ (AW1=1) operations. From an electrical level point of view, the four transistors of the driver (Mtp1, Mtn1, Mtp2, and Mtn2) are turned ON. Thus, there is a resistive short between VDD and GND nodes.

In order to define the level of BL and BLB during this scenario, the structure and sizing of the write driver have to be deeply analyzed. 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 the write driver, 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 write driver which is helped by its pre-charge circuit. However, as previously mentioned, the N plan (Mtn1 and Mtn2) is stronger than the P plan (Mtp1 and Mtp2), thus 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 logic '0' during the FP3 simulation seen in Fig. 5.6. 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 a logic '1' as the voltage level on node S was higher than the voltage level on node SB.

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.

5.3.3 March Test Solution Detecting SWDFs

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 are

$$w\mathrm{x}\ \ w\overline{\mathrm{x}}\ \ r\overline{\mathrm{x}} \tag{5.1}$$

where the two write operations are for sensitization of the fault and the read operation is for observation. x=0 (resp. $\overline{\mathrm{x}}$=1) corresponds to the detection of Df5 (resp. Df6). Assuming that these three operations have to be applied on the same core-cell, it is easy to create a specific March test to detect SWDFs as presented in Van de Goor et al. (2004). March WDm (4N complexity) and March WDw (8N complexity) are possible algorithms. 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. Consequently, possibilities to embed (with additional March elements) or find (with modifications based on the DOFs of March tests) successive operations required for the detection of SWDFs in existing March algorithms have to be developed.

Let us consider the basic view of an SRAM array as shown in Fig. 5.7 in which the write driver (WD) is shared by four columns. As the goal is to detect a possible malfunction of the write driver, it is not necessary to act the three operations presented in Equation 5.1 on the same core-cell. In fact, the first write operation (wx) 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 operation ($w\overline{\mathrm{x}}$) on the same core-cell but, at least, act this write operation on a core-cell of the four columns that initially

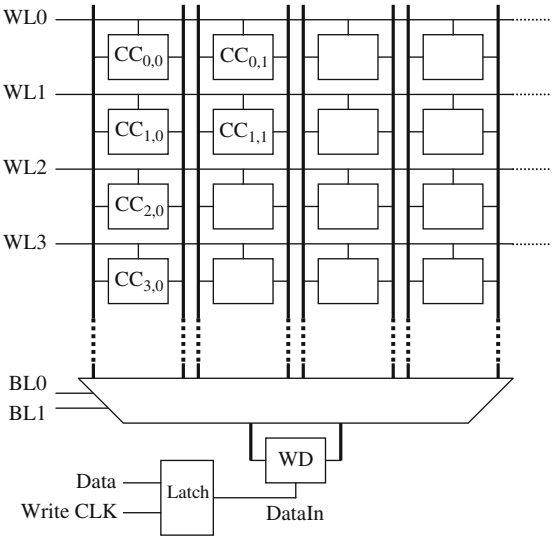


Fig. 5.7 Basic view of a part of an SRAM array

contains an opposite data to the data used for the write operation. Of course, the read operation ($r\overline{x}$) has to be performed on the last selected core-cell to control if the second write operation has been performed correctly. This statement makes the requirements presented in Equation 5.1 less stringent.

In addition, to further reduce the stringency of the required conditions to detect SWDFs, the driver control part has to be analyzed in detail. It is controlled by a *Write Enable* signal to perform the write operation with a certain data applied on the *DataIn* input (see Fig. 5.2). Moreover, as shown in Fig. 5.7, this data is latched, thus meaning that a logic '0' ('1') is captured in the latch for a $w0$ ($w1$) operation. Consequently, an important property is that when a $w0$ ($w1$) operation is acted by the write driver, this data (DataIn) remains stable in the latch as long as another write operation is not performed. For the detection of the SWDFs, the latch of the write driver captures the first data that has to be written. Thus, it is not necessary to act immediately the second write operation 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 that do not change the content of the faulty core-cell. Consequently, the resulting successive operations required to detect SWDFs become:

$$w\mathrm{x}\ \ (w\,\text{and/or}\,r)\ \ w\overline{\mathrm{x}}\ \ (w\,\text{and/or}\,r)\ \ r\overline{\mathrm{x}} \tag{5.2}$$

From these less stringent test conditions, the March algorithm must have the following requirements:

1. 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.
2. The presence of an $r1$ operation is necessary for observation of SWDFs due to Df5 and an $r0$ operation for those induced by Df6.

These two requirements can easily be found in many March algorithms. As shown in Ney et al. (2007a), the March C- is able to detect all SWDFs independently of the addressing order and data background. Figure 5.8 summarizes the March C- properties for SWDFs detection.

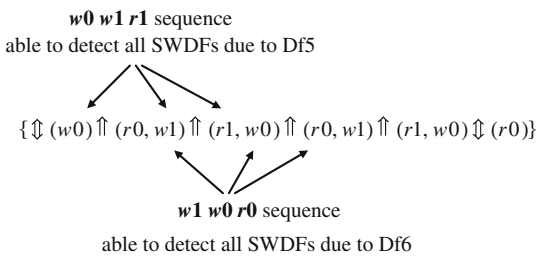


Fig. 5.8 Properties of March C- for SWDF detection

5.4 Analysis and Test of URWF/URDWF

In this section, the behavior of the write driver in the presence of defect Df9 is detailed. First, a FFM of the faulty behavior is presented. Next, electrical measurements are used to analyze the impact of Df9 on the behavior of the SRAM. As shown in Table 5.1, Df9 may induce two different dynamic behaviors, which are either a standard URWF or an URDWF. From this statement, comparisons of both URWF and URDWF are provided. Finally, it is shown that these faults can be detected by standard March algorithms.

5.4.1 Functional Fault Modeling of URDWF

In presence of Df9, an URDWF may occur. A $w0$ operation can be performed by the write driver, i.e., node AW0 (see Fig. 5.4) 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. However, Df9 prevents this pull-down and thus node AW0 remains at a logic '1' a certain time depending on the defect size. Therefore, the write driver continues to perform a $w0$ even if another operation has to be performed.

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

– FP1: <*ad1*($1w_0$), *ad2*($1r_1$)/*ad2*(0)/*ad2*(0)> A $w0$ is performed on a core-cell *ad1* containing a logic '1.' Then, an $r1$ is performed in another core-cell *ad2* (*ad1*≠*ad2*) belonging to the same bloc than *ad1*. This read operation returns a faulty logic '0' on the memory output and has inverted the core-cell content (from a logic '1' to a logic '0').
– FP2: <*ad1*($0w_0$), *ad2*($1r_1$)/*ad2*(0)/*ad2*(0)> This is the same description as for 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 (see Fig. 5.1):

– FP3: <*ad1*($0w_1$), *ad2*($0r_0$)/*ad2*(1)/*ad2*(1)> A $w1$ is performed on a core-cell *ad1* containing a logic '1.' Then, an $r0$ is performed in another core-cell *ad2* (*ad1*≠*ad2*) belonging to the same bloc than *ad1*. This read operation returns a faulty logic '1' on the memory output and the core-cell content has swapped (from a logic '0' to a logic '1').
– FP4: <*ad1*($1w_1$), *ad2* ($0r_0$)/*ad2*(1)/*ad2*(1)> This is the same description as for FP3, but this time, the $w1$ is performed on a core-cell containing a logic '1.'

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

$$\text{FP1} \equiv \text{FP2 and FP3} \equiv \text{FP4} \tag{5.3}$$

Furthermore, FP3 is the dual of FP1. So, the functioning is the same with opposite values. Consequently, the analysis of one of those is sufficient for a complete understanding of URDWFs.

5.4.2 Experiments

Hereafter, electrical simulations of the write driver in presence of Df9 are detailed. These simulations have been performed using a 65-nm SRAM technology.

5.4.2.1 URDWF Due to Df9

Waveforms in Fig. 5.9 present the resulting faulty behavior of the memory in presence of Df9 with typical PVT conditions (typical process, 1.2 V supply voltage, 27°C) and a defect size of about 500 kΩ. These electrical simulations have been performed using a 65-nm technology.

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. They initially contain a logic '1.' A $w0$ operation is first applied on CC_A. The pre-charge circuits are switched OFF. Node WD drives the logic '0' through BL to fight against the core-cell that contains a logic '1.' This means that the NMOS transistor (Mtn1 in

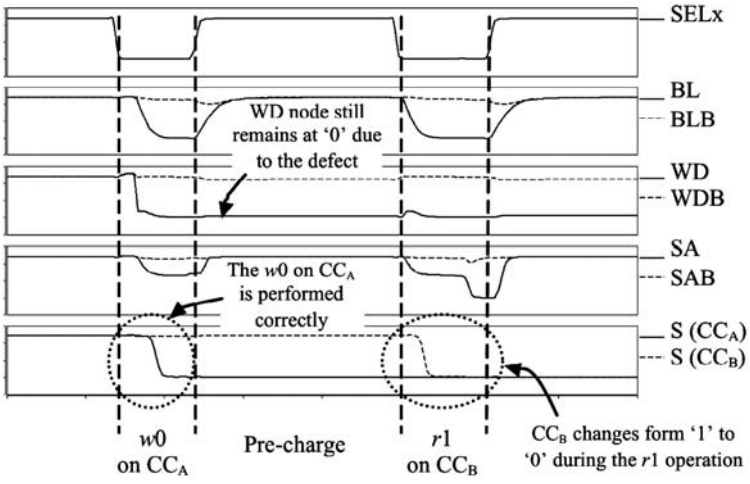


Fig. 5.9 Waveforms of FP1 simulation (Df9)

Fig. 5.4) has to be strong enough to impose the logic '0' on BL. The *w*0 operation is correctly performed on CC_A that swaps 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 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 logic '0' during the pre-charge operation in presence of defect Df9. Consequently, the *w*0 operation remains active.

Afterward, the second core-cell CC_B is selected for an *r*1 operation. In order to explain the resulting faulty behavior, it is important to remember the functioning of the sense amplifier. It allows taking a decision depending on the core-cell content (a 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 the presence of Df9, the fact that node WD remains at logic '0' makes that the differential voltage is incorrect and the *r*0 operation is erroneous. As seen in Fig. 5.9, node SA is at VDD and node SAB reaches a logic '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 logic '0' during the read operation, this level performs a *w*0 on CC_B thus inducing the swaps of the core-cell content from a logic '1' to alogic '0.'

To summarize the effect of Df9, the *r*0 operation on CC_B has two effects related to the fact that the write driver continues to perform a *w*0 during this read operation. First, the sense amplifier provides the data given by the write driver – a logic '0' in case of Df9. Secondly, CC_B is written to a logic '0.' So, the *r*0 operation is seen as a *w*0 operation.

5.4.2.2 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 the defect size (see Column 6 in Table 5.1) denoted to as Border 2 in Fig. 5.10. 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.

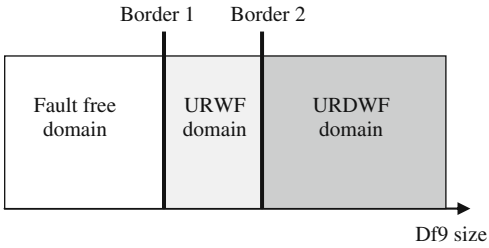


Fig. 5.10 Fault type vs. defect size

Once again, let us consider two core-cells (CC_A and CC_B) of the same group of columns controlled by the same I/O circuitry. Both core-cells initially contain a logic '1.' Waveforms in Fig. 5.11 present the faulty behavior of the memory in presence of Df9 with typical PVT conditions (typical process, 1.2 V supply voltage, 27°C) and a defect size of about 100 kΩ. These electrical simulations have been performed using a 65-nm technology.

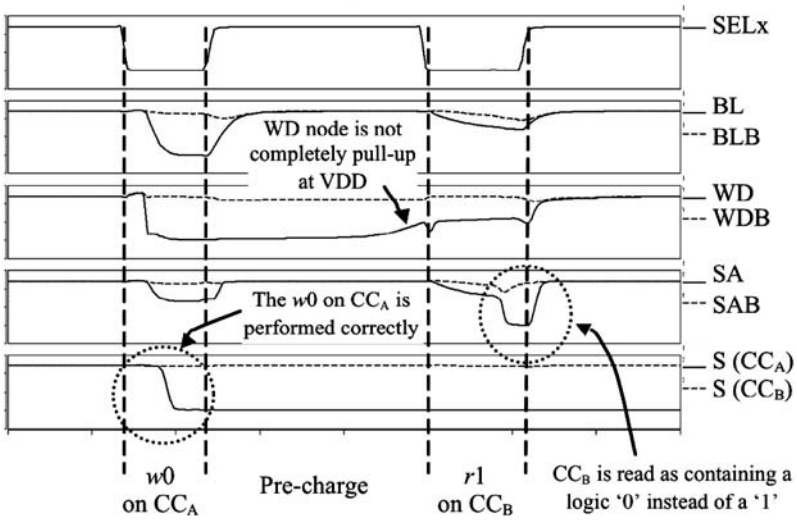


Fig. 5.11 An URWF due to Df9

The $w0$ operation performed on CC_A is correctly acted as the core-cell content swaps from a logic '1' to a logic '0.' Then, pre-charge circuits are switched ON. Compared to Fig. 5.7, this time node WD is not at logic '0' but rather is increasing. This is due to the fact that the NMOS transistor (Mtn1 in Fig. 5.4) is not fully saturated due to a lower Df9 size. Thus, it fights against the PMOS transistors of the pre-charge circuit. Then, at the beginning of the $r0$ operation performed on CC_B, the remaining voltage level on WD is not low enough to induce the faulty swap of the core-cell content. On the other hand, WD remains at a voltage level which is still low thus inducing that the sense amplifier badly translates the faulty differential voltage. This is shown in Fig. 5.11 where it can be seen that the content of core-cell CC_B does not swap (node S of CC_B still remains at a logic '1') but the logic data given by the sense amplifier is a logic '0' (node SA remains close to VDD and node SAB is at logic '0').

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.

5.4.3 March Test Solution Detecting URDWFs

As seen previously, defect 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:

$$w\mathrm{x}\, r\overline{\mathrm{x}} \tag{5.4}$$

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 Dilillo et al. (2005d). This study has shown that the March C- algorithm with a Fast_c addressing order is able to detect all URWFs. As the detection conditions of URWFs and URDWFs are the same, the March C- algorithm with a Fast_c addressing order is also able to detect all URDWFs.

5.5 Conclusion

This chapter has presented an analysis and the characterization of the effects induced by resistive-open defects that may occur in the write driver of SRAMs. It has been shown that two resistive-open defects may lead to a Slow Write Driver Fault (SWDF). This dynamic fault prevents the write control part to correctly decide between $w0$ and $w1$ operations. Moreover, it has been shown that a resistive-open defect may lead to a new type of dynamic behavior. 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. Electrical simulations using a 65-nm technology have been reported to give a complete characterization of these dynamic faults and especially to highlight differences between URDWF and the standard Un-Restored Write Fault (URWF) model. Finally, it has been shown that the March C- algorithm is able to detect all SWDFs as well as all URWFs/URDWFs by using a Fast_c addressing order.

Chapter 6
Resistive-Open Defects in Sense Amplifiers

6.1 The SRAM Sense Amplifier

This section describes the structure of each sense amplifier in the I/O circuitry. It first provides a global view of the memory including the I/O circuitry and then it details the functioning of the sense amplifier.

6.1.1 Sense Amplifier Within the I/O Circuitry

As already shown in Chapter 5, an SRAM I/O circuitry is composed of write drivers and sense amplifiers and is used to control or observe BLs and BLBs during the write and read operations of a given core-cell. A simplified view of the memory structure in which only sense amplifiers are represented (write drivers are not represented for the sake of clarity) is shown in Fig. 6.1. From this figure, 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 Fig. 6.1, 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 the selected column are propagated towards each SA_i and SAB_i nodes ($0 \leq i \leq k$). Then, the sense amplifier connected to the targeted core-cell is activated by its signal $SAON_i$ (all the others sense amplifiers 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 (DataOut) and is made of a latch. Its behavior is described in the next sub-section. 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.

6.1.2 Operation Mode of the Sense Amplifier

The transistor view of the considered sense amplifier is presented in Fig. 6.2. As previously mentioned, before every read operation, BL and BLB as well as SA

A. Bosio et al., *Advanced Test Methods for SRAMs*,
DOI 10.1007/978-1-4419-0938-1_6, © Springer Science+Business Media, LLC 2010

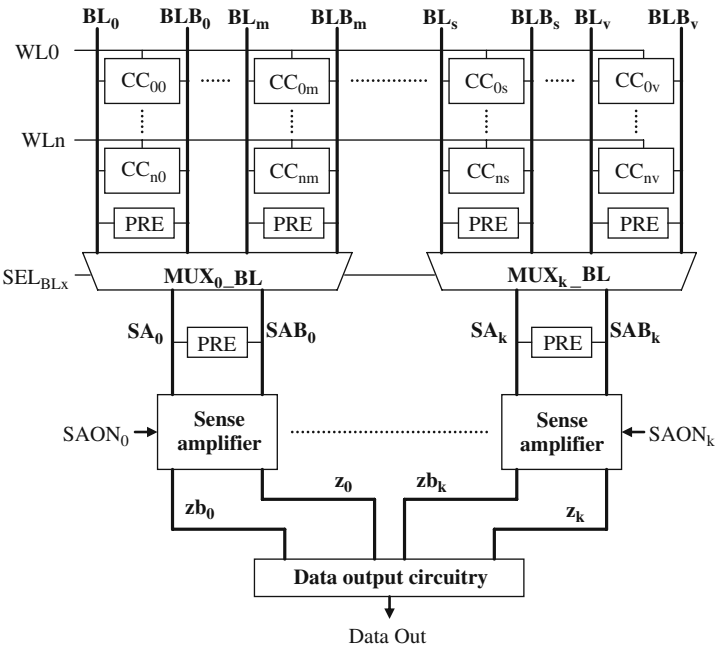


Fig. 6.1 Simplified SRAM view including sense amplifiers

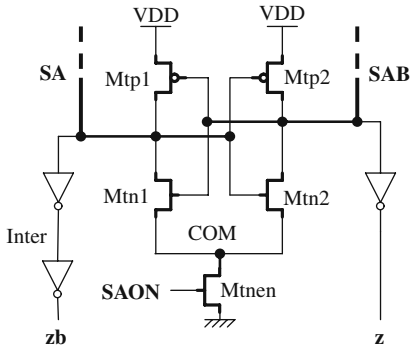


Fig. 6.2 Transistor view of a sense amplifier

and SAB are pre-charged at VDD. A read operation begins with the selection of the targeted core-cell. The access time allows one of the two bit lines (BL for an *r*0, BLB for an *r*1) to be discharged of about 10% of VDD (about 100 mV for an SRAM designed with a 65-nm technology).

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

- ΔBL$\approx$+100 mV for an $r1$ operation $\Rightarrow$ SA = 1 and SAB = 0
- ΔBL$\approx$–100 mV for an $r0$ operation $\Rightarrow$ SA = 0 and 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 Fig. 6.2), thus helping the discharge of the two nodes. However, the node with a lower voltage value (SA for an $r0$ and SAB for an $r1$) discharges faster than the other one, thus turning on the corresponding PMOS transistor (Mtp2 for an $r0$ and Mtp1 for an $r1$) after a certain time. Then, the voltage level on node SAB increases and finally reaches VDD. This prevents the opposite PMOS transistor (Mtp1 for an $r0$ and Mtp2 for an $r1$) to be turned ON, and the NMOS transistor (Mtn1 for an $r0$ and Mtn2 for an $r1$) to remain turned ON. At the end, the following configurations are obtained for a read operation performed on a core-cell belonging to group i ($0 \leq i \leq k$) of core-cells controlled by the same sense amplifier:

- for an $r0$ operation:

$$SA_i = 0 \text{ and } SAB_i = 1$$
$$z_i = 0 \text{ and } zb_i = 0$$

- for an $r1$ operation:

$$SA_i = 1 \text{ and } SAB_i = 0$$
$$z_i = 1 \text{ and } zb_i = 1$$

Note that all the other nodes SA_j and SAB_j (with $j \neq i$) 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. The structure of the data output circuitry is generally a latch with the functioning reported in the truth table of Table 6.1.

Table 6.1 Truth table of the data output circuitry

z	zb	DataOut
0	0	0
1	0	Memory state
0	1	Memory state
1	1	1

For an $r0$ operation on a core-cell belonging to group i, z_i and zb_i are at logic '0,' thus implying DataOut to be pulled-down. For an $r1$ operation, both z_i and zb_i are at logic '1,' implying DataOut to be pulled up. Note that when no read operation is performed or during the pre-charge operations, SA and SAB remains at VDD, thus implying z = 1 and zb = 0. With such a configuration, the DataOut signal remains stable at the logic data stored previously ('Memory state' in Table 6.1).

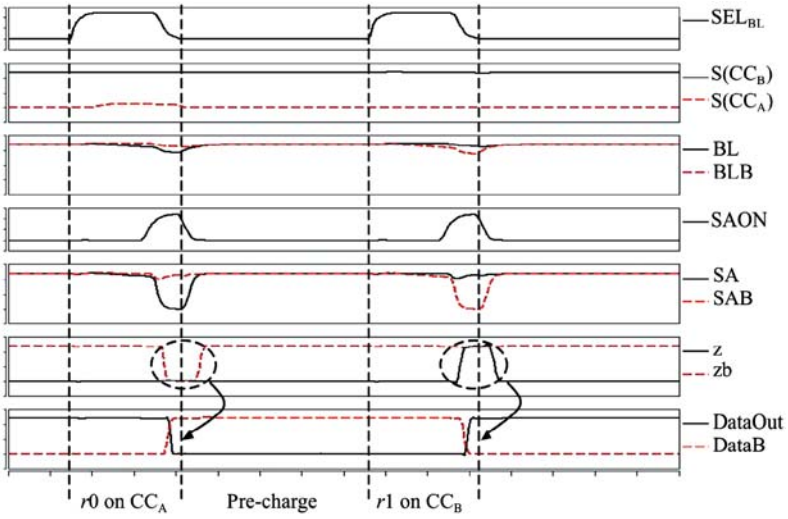


Fig. 6.3 Fault-free operation of a sense amplifier ($r0$ and $r1$ operations)

Waveforms presented in Fig. 6.3 show the correct operation of a sense amplifier during two consecutive read operations. Especially, an $r0$ followed by an $r1$ on two different core-cells (CC_A and CC_B) sharing the same sense amplifier are performed. S_{CCA} and S_{CCB} are the state values of each core-cell. These waveforms were obtained from typical PVT conditions (Typical process, 1.2 V supply voltage and 27°C temperature).

The simulation starts with an $r0$ 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, signal SAON is activated and the sense amplifier detects this weak differential voltage between SA and SAB. SA is fully discharged while SAB remains at VDD, so that nodes z and zb are set to logic '0.' With such logic values, node DataOut is pulled down (*c.f.* Table 6.1).

Then, pre-charge circuits are switched ON. All the lines (BL, BLB, SA, and SAB) are therefore forced to VDD. Note that DataOut remains stable at logic '0,' which corresponds to the last stored data (*c.f.* Table 6.1).

The next operation is an $r1$ 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 logic '0.' Then, nodes z and zb are set to a logic '1,' thus implying the pull-up of node DataOut.

6.2 Analysis of Resistive-Open Defects in the Sense Amplifier

This section first shows how and where resistive-open defects can be injected in the sense amplifier. Next, the impact of each defect is qualitatively analyzed and the

corresponding fault modeling is presented. Finally, the fault modeling is validated with electrical simulations.

6.2.1 Defect Location

As shown in Fig. 6.4, nine resistive-open defects (Df1–Df9) will be considered. The remaining defect locations can actually be deduced from these nine locations due to the symmetry of the sense amplifier structure.

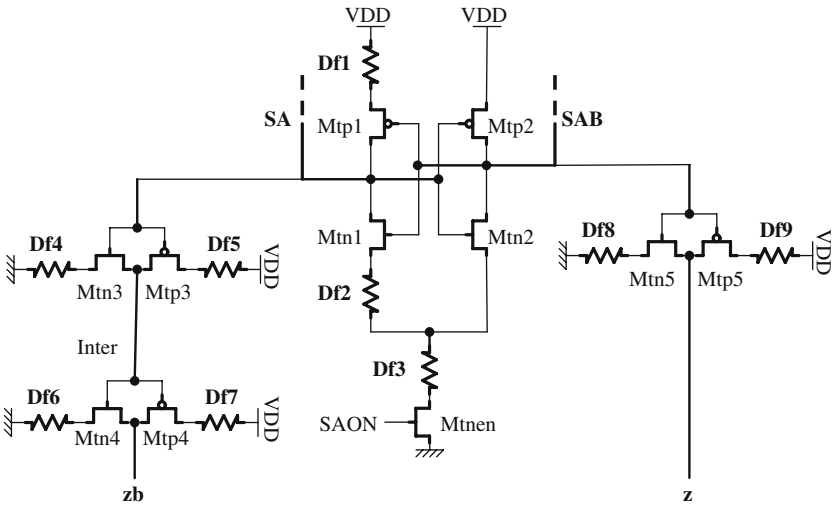


Fig. 6.4 Resistive-open defect injection in the sense amplifier

6.2.2 Defect Incidence Analysis

The behaviors produced by each defect in the sense amplifier are described below:

- *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 a data with an opposite value to that stored in the read core-cell. For this defect, a logic '1' is observed when an $r0$ operation is acted as the pull-down of node SA cannot correctly be performed. This faulty behavior can be modeled as an Incorrect Read Fault (IRF).
- *Defect Df3:* During a read operation, SA (for an $r0$) or SAB (for an $r1$) node is normally driven to logic '0' when the sense amplifier is activated by its SAON signal. However, in the 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 provides the logic data previously stored. Such faulty behavior is modeled as a dynamic 2-cell Incorrect Read Fault of type 1 (d2cIRF1).
– *Defects Df4 to Df9:* These defects prevent the pull-up or the pull-down of nodes z and zb. Two successive read operations with two different values on two distinct core-cells are therefore not possible. This faulty behavior is modeled as a dynamic 2-cell Incorrect Read Fault of type 2 (d2cIRF2).

Table 6.2 presents a summary of the fault models identified for each injected resistive defect.

Table 6.2 Fault models identified for each injected defect

Defect	Fault model
Df1	–
Df2	IRF
Df3	d2cIRF1
Df4	d2cIRF2
Df5	d2cIRF2
Df6	d2cIRF2
Df7	d2cIRF2
Df8	d2cIRF2
Df9	d2cIRF2

Definitions of fault models reported in Table 6.2 are the following:

Incorrect Read Fault (IRF) (Van de Goor 1998): A sense amplifier is said to have an IRF if a read operation performed on a core-cell returns an incorrect logic value, while keeping the correct stored value in the read core-cell.

dynamic two-cells Incorrect Read Fault of type 1 (d2cIRF1) (Ney et al.2007b): 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)* (Ney et al. 2009): A sense amplifier is said to have a d2cIRF2 if it is only able to perform an $r0$ or an $r1$ operation. As for d2cIRF1, this is a two-cell fault model as it requires two read operations on two distinct core-cells.

Resistive-open defects in a sense amplifier of an SRAM can be modeled as static faults (IRF) as well as dynamic faults (d2cIRF of type 1 and type 2). The next sections are dedicated to the study of these dynamic faults.

6.2.3 Simulation Set-Up and Results

In Ney et al. (2009), the authors have examined the whole range of operating conditions with the aim of determining those that maximize the fault detection probability. Simulations have been performed using a 65-nm technology by applying a number

of different test patterns (according to the expected fault models defined above) with a defect size that ranges from a few ohms up to several megaohms and for different PVT parameters:

- *Process Corner*: slow, typical, fast, fast n/slow p, slow n/fast p
- *Voltage Supply*: 1.08 V, 1.2 V, 1.32 V
- *Temperature*: –30°C, 27°C, 110°C

Table 6.3 presents the conditions for maximum fault detection (i.e., the PVT conditions), the minimum detected resistance value and the observed fault model.

Table 6.3 Summary of worst-case PVT corners for the defects of Fig. 6.3 and corresponding minimum detected resistance and fault model

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

The first column (Dfi) indicates the defect location in the sense amplifier according to Fig. 6.3. The following three columns correspond to the electrical parameters which maximize the fault detection. The fifth column provides the minimum defect size that induces a faulty behavior. Finally, the last column gives the observed fault models.

6.3 Analysis and Test of d2cIRF1

As shown in the previous section, a d2cIRF1 can be produced by a resistive-open defect (Df3 in Fig. 6.4). This section first presents a Functional Fault Modeling (FFM) of such a defect. Next, the conditions required to detect all d2cIRF1 as well as a possible March test solution are detailed.

6.3.1 Functional Fault Modeling of d2cIRF1

In presence of Df3, a d2cIRF1 may occur depending on the size of the defect. During a read operation, SA (for an $r0$) or SAB (for an $r1$) is normally driven to logic '0' when the sense amplifier is activated by its SAON signal. However, in the presence of Df3, this operation cannot be performed as the sense amplifier remains disabled. Consequently, the data output circuitry does not change its value and outputs the

logic data previously stored. At this point the question is, How this faulty behavior can be sensitized and observed?

A straightforward solution consists in initializing the data output circuitry by performing a read operation with a given sense amplifier (Sense_1). This operation is denoted as rx with x $\in \{0, 1\}$. The data output circuitry is therefore initialized at the 'x' logic value. Then, another sense amplifier (Sense_2), sharing the same data output circuitry, is selected to perform a read operation with an opposite data. 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 a logic 'x' instead of providing a logic '$\bar{x}$.' 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 which is independent of the memory configuration consists in performing the two read operations, rx and r$\bar{x}$, on the same sense amplifier. This time, DataOut 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 Fault Primitive (FP):

– FP: <*ad1*(xr_x), *ad2*($\bar{x}r_{\bar{x}}$)/*ad2*($\bar{x}$)/*ad2*(c)> An rx operation is performed on a first core-cell with an address *ad1*. Then, an r$\bar{x}$ operation is performed with the same sense amplifier in another core-cell with an address *ad2* (*ad1*≠*ad2*). The node DataOut still remains at a constant logic value 'c' during both read operations.

6.3.2 Experiments

Hereafter, electrical simulations of the sense amplifier are detailed. These simulations have been performed using a 65-nm technology. Waveforms in Fig. 6.5 present the faulty behavior of the memory in presence of Df3. They were obtained with typical PVT conditions (typical process, 1.2 V supply voltage, 27°C temperature) and a defect size of about 10 kΩ.

This simulation reports the behavior of 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 the data output circuitry (DataOut) initialized at a logic '0.'

An 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. The data output circuitry is in a memory state and hence does not change its output value, which remains at logic '0.' The fault is not observed as the read data (a logic '0' in this case) is the same as that initially stored in the data output circuitry.

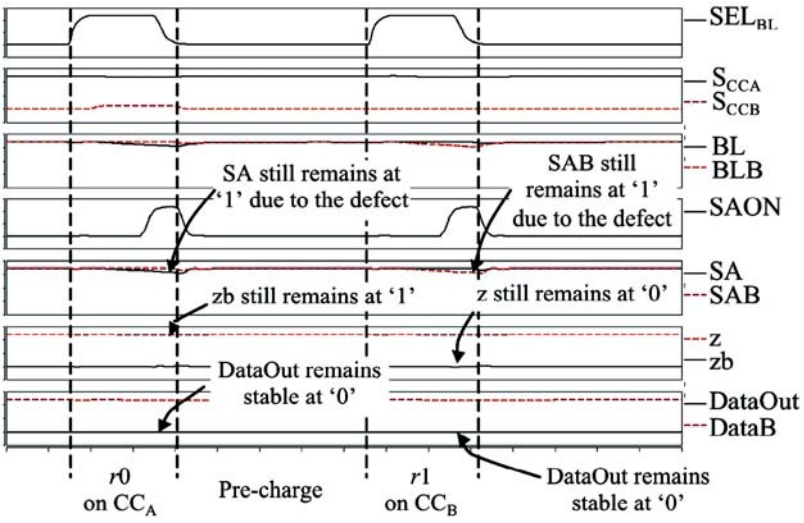


Fig. 6.5 Waveforms of <$CC_A(0r_0)$, $CC_B(1r_1)/CC_B(1)/CC_B(c)$> simulation (Df3)

Then, a second read operation is performed with an $r1$ 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 remains 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 DataOut 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, two read operations with opposite data on the same sense amplifier have to be applied to guarantee the d2cIRF detection independently of the initial DataOut value.

6.3.3 March Test Solution Detecting d2cIRF1s

As shown previously, a d2cIRF1 may occur in the 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:

$$r\mathrm{x}\, r\bar{x} \tag{6.1}$$

where both read operations have to be obviously performed on two distinct core-cells sharing the same sense amplifier. As shown in (Ney et al. 2007b), three remarks can be formulated about the possible modifications allowed on this sequence of sensitization operations.

Remark 1: When an *r*x operation is performed by a sense amplifier, the DataOut node of the corresponding data output circuitry remains stable as long as an $r\bar{x}$ operation is not performed by a sense amplifier that shares the same data output circuitry. Consequently, any type of write operation is allowed between the two read operations of the detection sequence.

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

Remark 3: If an $r\bar{x}$ operation is performed with another sense amplifier that does not share the same data output circuitry with the targeted one, then the DataOut 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.

As presented in Fig. 6.6, these three remarks allow a less stringent sequence of sensitization conditions for the d2cIRF1 detection. From this statement, it is easy to create a specific March test to detect d2cIRF1s. However, as previously mentioned, it is more interesting to obtain a March test that covers a larger set of fault models rather than only d2cIRF1s. Hereafter, the ability of March C- to detect all d2cIRFs is analyzed. To ease the discussion below, the March C- is defined again in Fig. 6.7.

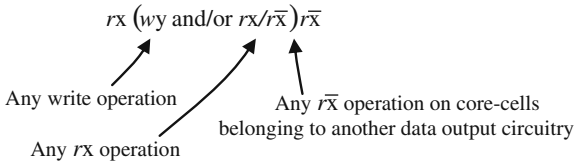


Fig. 6.6 Relaxed constraints to detect d2cIRF1

$$\{\Updownarrow(w0)\Uparrow(r0,w1)\Uparrow(r1,w0)\Downarrow(r0,w1)\Downarrow(r1,w0)\Updownarrow(r0)\}$$
M0 M1 M2 M3 M4 M5

Fig. 6.7 March C- algorithm

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 an $r0$ 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).

Then, March element M2 is applied using the same addressing order as M1. The first targeted core-cell is selected for an $r1$ 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. The application of March elements M1/M2 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).

Consequently, a straightforward solution would consist in applying the two read operations required for d2cIRFs detection in a March element. The proposed solution consists in using the modifications of the March C- presented in Dilillo et al. (2004a). In this chapter, the authors have proposed a new March test called March iC- (Fig. 6.8) for ADOFs (Address Decoder Open Faults) detection. The particularity of this new March test 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 a 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 formerly detected by the March C-.

$$\underset{\text{M0}}{\{\Updownarrow (w\text{Av})} \underset{\text{M1}}{\Uparrow (r\text{Av},w\text{A}\overline{\text{v}})} \underset{\text{M2}}{\Uparrow (r\text{A}\overline{\text{v}},w\text{Av})} \underset{\text{M3}}{\Downarrow (r\text{A}\overline{\text{v}},w\text{Av})} \underset{\text{M4}}{\Downarrow (r\text{Av},w\text{A}\overline{\text{v}})} \underset{\text{M5}}{\Updownarrow (r\text{A}\overline{\text{v}})\}}$$

Fig. 6.8 March iC- algorithm

Using the concept of alternated data of the March iC-, a good addressing order has to be found to insure the detection of all d2cIRF1s. Let us consider element M1 and $v = 0$. The successive operations applied at different addresses are

$$\underset{\text{Add1}}{(r0,w1)}\ \underset{\text{Add2}}{(r1,w0)}\ \underset{\text{Add3}}{(r0,w1)}\ \underset{\text{Add4}}{(r1,w0)}$$

At this point, there are many possibilities to obtain the sequence of sensitization. But the simplest solution is to address a core-cell that uses a sense amplifier with Add1 and another core-cell that uses the same sense amplifier with Add2. Consequently, the couple ($r0$, $r1$) is acted with a $w1$ in between that does not mask the fault 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 (Ney et al. 2007b).

6.4 Analysis and Test of d2cIRF2

This section details the behavior of the sense amplifier affected by a d2cIRF2. As previously, a FFM of the faulty behavior by using FPs is first provided. Next, electrical measurements to analyze the impact of a d2cIRF2 on the SRAM behavior are reported. Finally, a possible March test solution to detect d2cIRF2s is presented.

6.4.1 Functional Fault Modeling of d2cIRF2

In the presence of defects Df4–Df9 a d2cIRF2 may occur. These defects can be organized into two groups as follows:

- 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 6.1). To sensitize defects of Group 1, nodes z and zb must be set to a logic '0.' This configuration corresponds to an $r0$ operation. Consequently, detection of defects belonging to Group 1 requires an $r0$ operation to initialize z and zb nodes at logic '0,' followed by an $r1$ operation for the fault 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 an $r1$ operation to initialize z and zb nodes at logic '1,' followed by an $r0$ operation for the fault sensitization.

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

– FP1: <*ad1*($0r_0$), *ad2*($1r_1$)/*ad2*(1)/*ad2*(0)> An $r0$ is performed on a first core-cell with an address *ad1*. Then, an $r1$ is performed in another core-cell with an address *ad2* sharing the same sense amplifier (*ad1*≠ *ad2*). A logic '0' is read on node DataOut instead of a logic '1.' This FP is related to defects of Group 1.
– FP2: <*ad1*($1r_1$), *ad2*($0r_0$)/*ad2*(0)/*ad2*(1)> An $r1$ is performed on a first core-cell with an address *ad1*. Then, an $r0$ is performed in another core-cell with an address *ad2* sharing the same sense amplifier (*ad1*≠ *ad2*). A logic '1' is read on node DataOut instead of a logic '0.' This FP is related to defects of Group 2.

6.4.2 Experiments

Waveforms in Fig. 6.9 present the faulty behavior of the memory in presence of Df4. They were obtained with typical PVT conditions (typical process, 1.2 V supply voltage, 27°C) and a defect size of about 500 kΩ. This simulation, obtained with a 65-nm technology, considers 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 DataOut initialized at a logic '1' value.

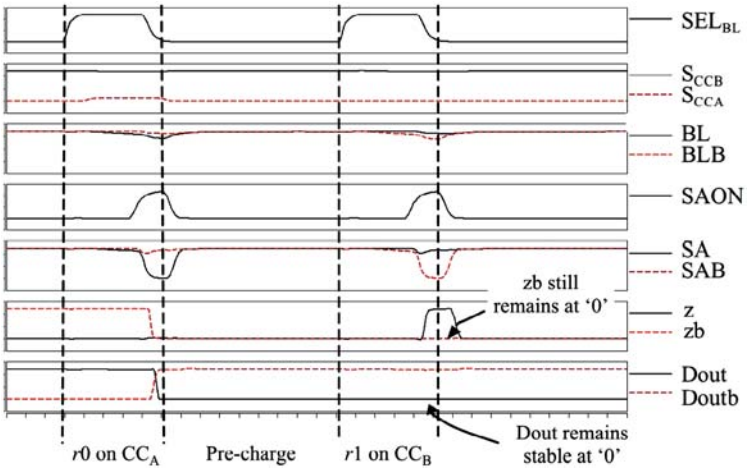


Fig. 6.9 Waveforms of <$CC_A(0r_0)$, $CC_B(1r_1)$/$CC_B(1)$/$CC_B(0)$> simulation (Df4)

An $r0$ operation is first applied on CC_A. BL node is slightly discharged (about 100 mV) 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 DataOut is at 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 node is not pulled up and thus remains at logic '0.'

A second read operation is then performed with an $r1$ 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 6.1). Hence DataOut provides a logic '0' instead of a logic '1.'

Note that electrical simulations of defects Df7 and Df9 are not reported here as they lead to the same behavior than those for defect Df4. Similarly, faulty behavior in presence of Df5, Df6, and Df8 can be obtained by duality.

6.4.3 March Test Solution Detecting d2cIRF2s

As shown previously, a d2cIRF2 may occur in presence of defects Df4, Df5, Df6, Df7, Df8, and Df9. Such a faulty behavior is sensitized and observed with specific sequences of read operations. These sequences are defined as follows:

- $r0r1$ for defects belonging to Group 1,
- $r1r0$ 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, less stringent detection sequences can be found, i.e., allow additional read or write operations between the two read operations required for detecting d2cIFR2. However, as defects impact the pull-up or the pull-down of nodes z and zb, read or write operations may mask the fault effect. For a complete understanding, waveforms in Fig. 6.10 present the memory operation in the presence of Df4 using worst case conditions (process: slow, voltage: 1.08 V, temperature: –30°, and Df4 = 140 kΩ). As shown in Table 6.3, with these conditions the memory is affected by a d2cIRF2 when an $r0$ operation is immediately followed by an $r1$ operation. To confirm the fact that two read operations must be applied sequentially, the memory operation has been simulated by applying the following

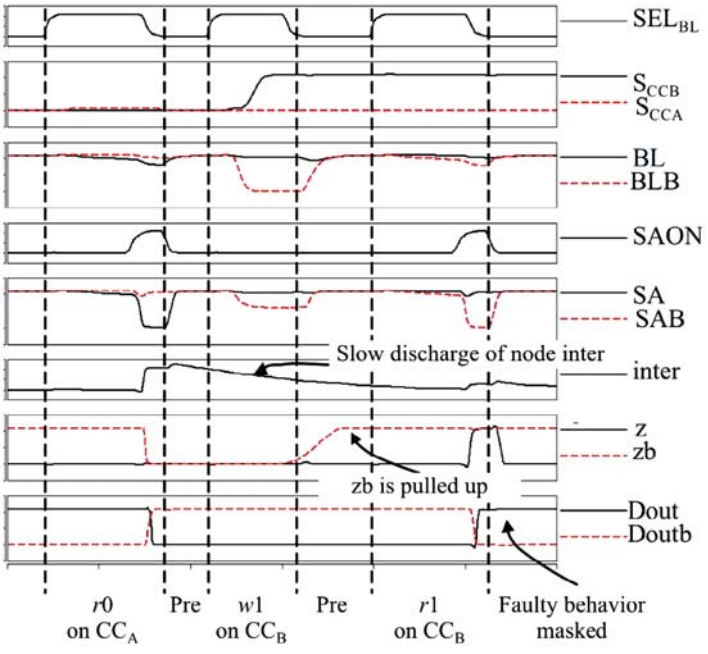


Fig. 6.10 Waveforms of $CC_A(0r_0)$, $CC_B(0w_1r_1)$ simulation (Df4)

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 Fig. 6.10.

First an $r0$ 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 DataOut provides a logic '0.'

Pre-charge circuits are then 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 defect Df4, node Inter (see Fig. 6.2) is not correctly pulled down. Consequently, zb remains at logic '0.'

Then, a $w1$ 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.'

Finally, a second read operation ($r1$) is applied on CC_B which contains a '1.' The faulty behavior of the sense amplifier is masked as node zb has reached VDD before the beginning of the read operation.

Consequently, the March algorithm must contain two successive read operation with opposite data value to insure the detection of all d2cIRF2s. The March iC- algorithm described previously is able to detect such faulty behavior. In fact, by considering element M5 (see Fig. 6.8), the succession of operation applied at different addresses is:

$$\begin{array}{cccccc} (r0) & (r1) & (r0) & (r1) & \ldots \\ \text{Add1} & \text{Add2} & \text{Add3} & \text{Add4} & \ldots \end{array}$$

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.

Based on these statements, 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. Other 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.

6.5 d2cIRF1 vs. d2cIRF2

Sections 6.3 and 6.4 have highlighted a new faulty behavior modeled by a dynamic two-cell Incorrect Read Fault (d2cIRF). There are two distinct ways to qualify this behavior:

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

These faulty behaviors represent resistive-open defects that affect the sense amplifier. They prevent it to operate correctly, especially when two successive read operations are performed. 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 6.4 summarizes which element of March iC- is able to detect d2cIRF1 and d2cIRF2.

Table 6.4 March iC- ability

	d2cIRF1	d2cIRF2
March iC-element	M1–M5	M5

6.6 Conclusion

This chapter has proposed the analysis and the characterization of the effects induced by resistive-open defects that may occur in the sense amplifiers of SRAMs. It has been shown that some resistive-open defects may lead to new types of dynamic behaviors. These faulty behaviors have been modeled as a d2cIRF1 and a d2cIRF2. Such fault models are a consequence of failures in the sense amplifier which prevent 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). Electrical simulations have been reported to give a complete understanding of such faulty behaviors. Moreover, it has been shown that the March C- with a specific data (alternated data value) and a specific addressing order (line or column) is able to detect all d2cIRFs that may affect the sense amplifiers of an SRAM. It is also important to notice that these modifications do not change the ability of March C- to detect the former targeted faults (stuck-at, transition, coupling, etc.).

Chapter 7
Faults Due to Process Variations in SRAMs

7.1 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 (Borkar et al. 2003). These atomic-level 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 deviations. On the other hand, local variations, resulting from mismatches in parameters of similar transistors (threshold voltage, V_t, 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_t, which is the main source of deviation due to random dopant effect (Bhavnagarwala et al. 2001), has been studied in Borkar et al. (2003) and Bastian et al. (2007).

In the following sections, the simulation flow used for mismatch injection is first presented. Then, an analysis of read and write operations to determine which transistors of the core-cell are candidates for V_t mismatch injection is detailed. Finally, achieved simulation results and test requirements for effective mismatch detection are presented.

A. Bosio et al., *Advanced Test Methods for SRAMs*,
DOI 10.1007/978-1-4419-0938-1_7,

7.1.1 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:

$$I_{ds} = \frac{1}{2} \times k \times \frac{W}{L} \times \left(V_{gs} - V_t\right)^2 \tag{7.1}$$

where

$$V_t = V_{t0} + K_{be} \times \left(\sqrt{|V_{sb}| + 2\Phi_F} - \sqrt{2\Phi_F}\right) \tag{7.2}$$

and

$$k = \mu \times C_{ox} \tag{7.3}$$

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

In this study, only threshold voltage mismatches are considered as they are the main sources of deviation due to random dopant effect (Bhavnagarwala et al. 2001). 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_t mismatches has been simulated with the following varying parameters:

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

V_t mismatch varies from 0σ up to $|6\sigma|$. 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.

7.1.2 Mismatch Sensitivity During Read/Write Operations

V_t 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_t mismatch injection, this sub-section presents a complete analysis of write and read operations.

For write operations, only V_t mismatches that reduce the core-cell transistor conductivity are considered. Let us consider the core-cell presented in Fig. 7.1 in which the core-cell originally stores a logic '1.' Node S is at VDD and node SB at GND. Remember that to write a logic '0' into this core-cell, BLB line remains at VDD, BL line is lowered to GND, and the core-cell is selected by applying VDD on WL. Operating devices and current flows during this $w0$ operation are illustrated in Fig. 7.1. 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. In the same way, it can be shown that Mtp2 and Mtn1 in addition to pass-transistors are used for a $w1$ operation.

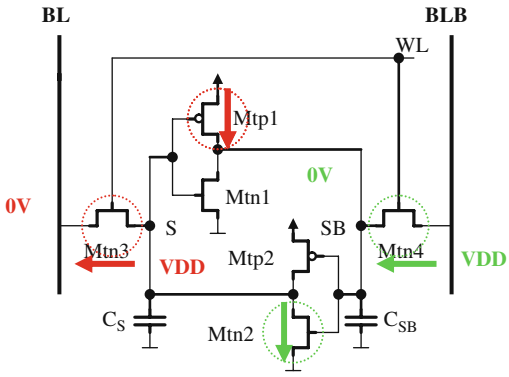


Fig. 7.1 Core-cell currents whose weakness is critical for a $w0$ operation

Waveforms of the different currents and voltage levels induced by the $w0$ operation are reported in Fig. 7.2. These curves show that the voltage at node S reaches VDD before node SB. Thus, node S controls the $w0$ operation. Conversely, node SB will control the $w1$ operation. Consequently, V_t 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 $w1$ operation).

In the same way, an analysis of which transistors of the core-cell are involved during a read operation is presented. In this case, not 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 core-cell has stored a logic '0.' Operating devices and current flows during this read operation are illustrated in Fig. 7.3. 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, 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 nodes SB and BLB are the same, no current flows and transistors Mtp1 and Mtn4 will maintain the VDD

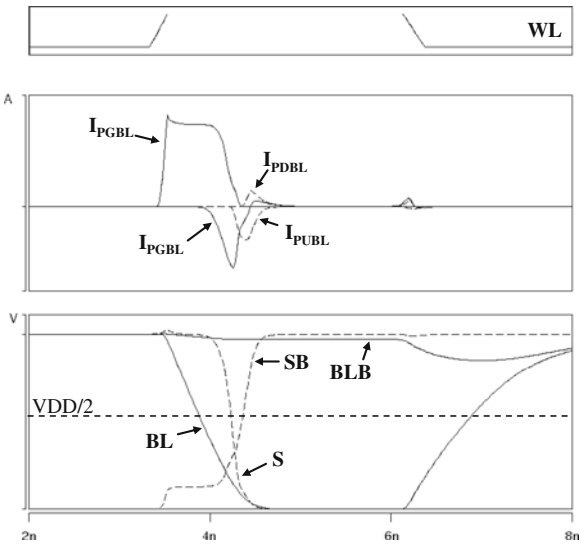


Fig. 7.2 Currents and voltages during a *w*0 operation

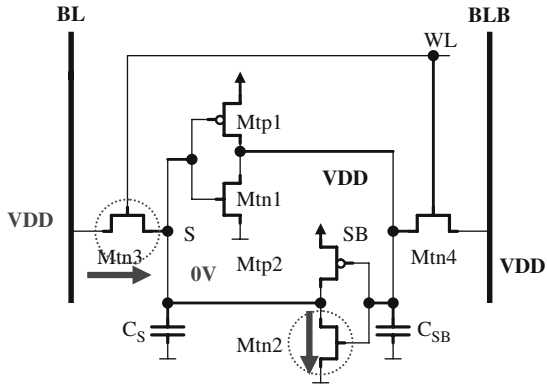


Fig. 7.3 Core-cell currents whose weakness is critical for an *r*0 operation

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.

Waveforms of currents and voltages involved during an *r*0 operation are presented in Fig. 7.4. At the end of the *r*0 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.'

This analysis demonstrates that V_t mismatches on Mtn3 and/or Mtn2 transistors will have an impact on the *r*0 operation (respectively, Mtn4 and/or Mtn1 transistors for an *r*1 operation).

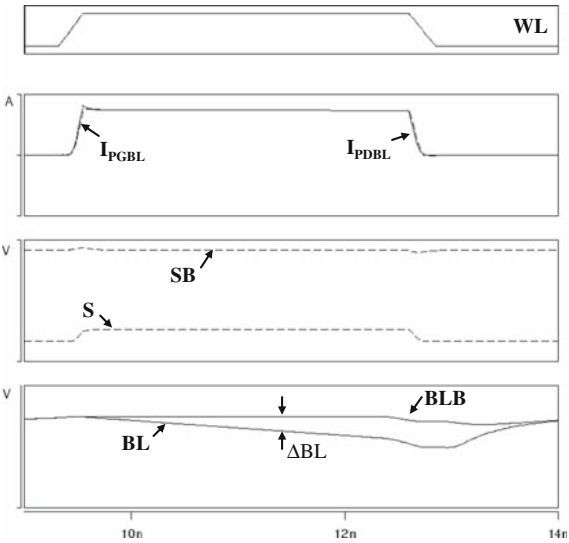


Fig. 7.4 Currents and voltages during an *r*0 operation

7.1.3 V_t *Mismatch Related Fault Models*

The previous sub-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, a mismatch injection in different locations of the core-cell can be done as presented in Fig. 7.5. The goal here is to provide a functional fault modeling of each mismatch configuration.

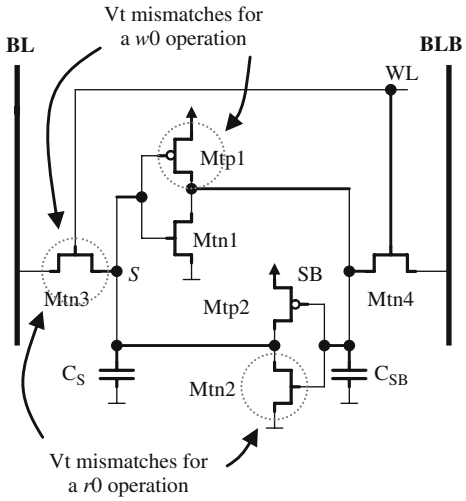


Fig. 7.5 Considered V_t mismatch locations for *w*0 and *r*0 operations

7.1.3.1 Result Overview

Simulations were performed considering single or double mismatch locations with identical V_t deviations (up to 6 σ). 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_t mismatch). Results are reported in Table 7.1.

Table 7.1 Results summary

Mismatch location	Mismatch size	PVT	Observed fault model
Mtn3	$\sim 4\ \sigma$	sf, 0.9 V, –40°C	TF
Mtn3 and Mtp1	$\sim 4\ \sigma$	sf, 0.9 V, –40°C	TF
Mtn3	$\sim 6\ \sigma$	fs, 0.9 V, 125°C	RDF
Mtn3 and Mtn2	$\sim 3\ \sigma$	fs, 0.9 V, 125°C	RDF
Mtn3	$\sim 3.8\ \sigma$	sf, 0.9 V, –40°C	dRDF

The first column gives the location of the V_t mismatch. The second column gives the minimum mismatch value that sensitizes the fault. The third column gives the PVT conditions that maximize the mismatch detection (i.e., worst case conditions). The last column gives the observed fault model.

The first result of these simulations is that PVT conditions that maximize the mismatch detection are always at low voltage (0.9 V). In fact, a V_t variation of 100 mV is proportionally higher for a supply voltage of 0.9 V than for a supply voltage of 1.5 V (see Equation 7.1). This first result shows that V_t mismatches 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 (Borri et al. 2005).

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 explained by the fact that V_t varies in a monotonous way with the temperature (linear relationship):

$$V_t(T) = V_{t_{nom}} + CTE \times (T_{nom} - T) \tag{7.4}$$

and

$$\frac{V_t(T)}{dT} = -CTE \leq 0 \tag{7.5}$$

It means that V_t is strictly decreasing when the temperature increases, so extreme corners maximize the detection of mismatches.

Faults observed are TF, Read Destructive Faults (RDF), and dynamic Read Destructive Faults (dRDF). Each fault is induced by a different combination of mismatches, sensitizing sequences, and PVT conditions. For example, Fig. 7.6 shows electrical simulations of a dRDF induced by V_t mismatches. This defective core-cell loses its content during the second at-speed $r0$ operation for a 3.8 σ deviation of V_t.

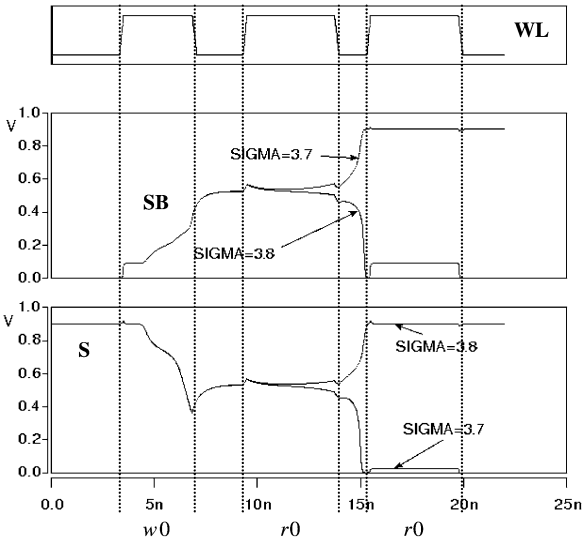


Fig. 7.6 V_t mismatch inducing dRDF (sf, 0.9 V, –40°C – Mtn3)

7.1.3.2 Test Requirements

It has been shown in the previous sub-section that a V_t mismatch induces different faulty behaviors which can be modeled by TF, RDF, and dRDF. Now, test requirements (algorithms and PVT conditions) needed to detect these fault models have to be analyzed.

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 (Al-Ars and Van de Goor 2001) or March C- with specific addressing order (Dilillo et al. 2004b), 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_t 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 (Borri et al. 2005). In addition, it has been shown in the previous sub-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.

7.2 Impact of Leakage Current of Core-Cell Pass-Transistors on the Read Operation

In SRAMs, the read operation is based on the detection/amplification of ΔBL generated by the selected core-cell. Thus, any further voltage difference between the two bit lines and not generated by the selected core-cell may be the cause of an incorrect output value. Referring to Fig. 7.7, let us consider the core-cell storing a logic '0,' and placed in the lower part of the diagram. Although this core-cell is not selected for the read operation, it does interact with the bit lines BL and BLB because of the leakage currents.

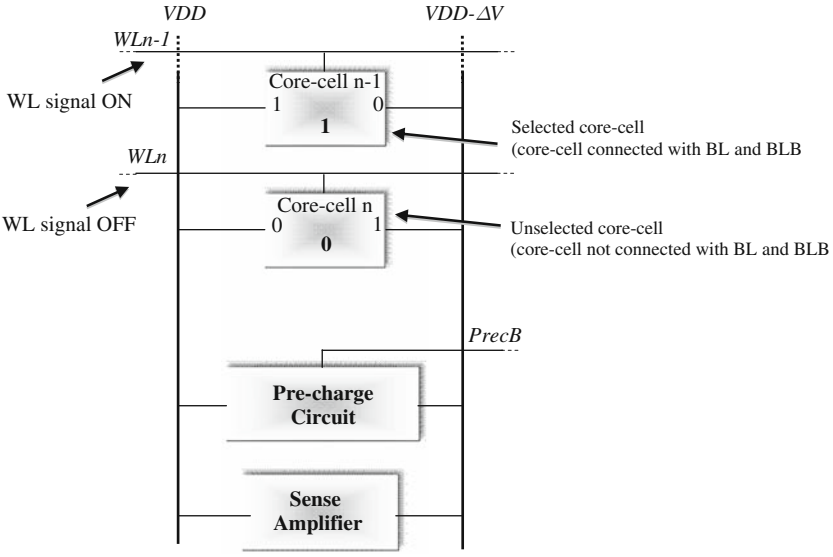


Fig. 7.7 Read operation in SRAMs

Figure 7.8 shows these interactions with some details. When the core-cell is unselected, the Word Line selection signal is low ($WLn = 0$) and the two pass-transistors are OFF. Although the pass-transistors are OFF, there is a certain amount of current that leaks through these two transistors thus discharging the bit lines.

In particular, the leakage current related to transistor Mtn3 has three components:

1. Subthreshold leakage current that flows from bit line BL, charged at VDD, to the substrate of transistor Mtn3 which is polarized at 0 V.
2. Gate leakage current that flows from bit line BL, charged at VDD, through the gate oxide, to the gate of transistor Mtn3 which is at 0 V (the core-cell is not selected). This current has been demonstrated to be significant for technologies under 90 nm (Mukhopadhyay et al. 2005).
3. Junction leakage current that flows from bit line BL, charged at VDD, to node S of the core-cell which is at 0 V.

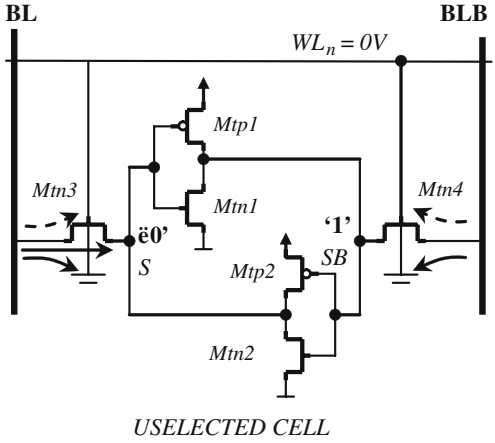


Fig. 7.8 Leakage currents through the pass-transistors in an unselected core-cell

In nanoscale technologies, the three components of the overall leakage current of a pass-transistor are becoming more important with the reduction of the transistor size and the thickness of the gate oxide (Taur and Ning 1998). On the right side of the core-cell, the leakage current related to transistor Mtn4 has two components, i.e., the subthreshold and the gate leakage currents (1) and (2), while the junction leakage current (3) does not occur because the core-cell node SB has the same voltage level (V_{SB} = VDD) than bit line BLB. The fact that the two nodes of the core-cell are at different logic levels induces that in an unselected core-cell the leakage currents from the bit lines through the two pass-transistors are not symmetric. In particular, the leakage is higher on the side with the core-cell node at logic '0,' e.g., on the left side as in Fig. 7.8. An SRAM column has hundreds of core-cells, and only one core-cell at a time can be selected for the read operation. Each unselected core-cell of the column interacts with the bit lines BL and BLB, because of the leakage currents. The leakage currents drawn by each unselected core-cell affect the voltage difference observed by the sense amplifier.

Figure 7.9 shows a memory column where one core-cell, storing a logic '1,' is selected for the read operation and the other core-cells are not selected and store various values. During the read operation, the selected core-cell discharges partially one of the two bit lines and the sense amplifier detects the voltage difference and amplifies it for the output. In the case depicted in Fig. 7.9, the selected core-cell partially discharges bit line BLB. Due to the leakage currents, the unselected core-cells in the column also give their contribution to the discharge of both bit lines BL and BLB in a different way, depending on the stored values. In particular, when the number of unselected core-cells storing a logic '0' is higher than the number of unselected core-cells storing a logic '1,' the overall effect is that bit line BL is more discharged than bit line BLB. This effect of leakage currents is opposite to the effect of bit line discharge due to the $r1$ operation on the selected core-cell. In other words, the sum of the effects of the asymmetric leakage currents changes

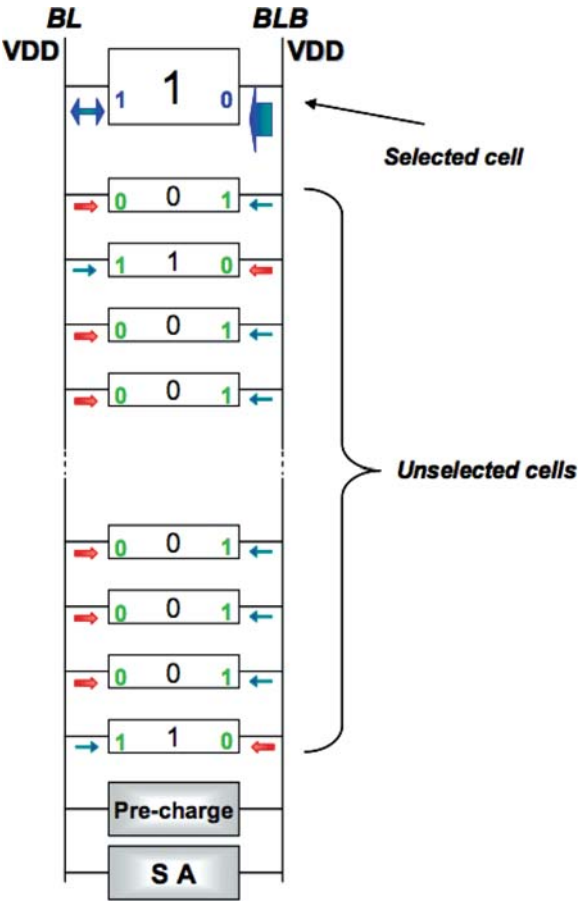


Fig. 7.9 The leakage effect in a memory column

the voltage difference between the two bit lines. A significant reduction of ΔBL increases the noise sensibility of the device and may lead to a faulty read operation which is modeled as a Leakage Read Fault (Dilillo et al. 2006c) defined as follows:

Leakage Read Fault (LRF): When in a memory column most core-cells store the same value x ∈ {0,1}, the leakage currents, through the pass-transistors of the unselected core-cells may affect the read operation in the core-cells storing the value $\overline{x}$, where $\overline{x}$ is the opposite of x: $\overline{x}$ is expected and x is read.

From the above definition, it is possible to define the generic FP modeling the faulty behavior of a LRF:

$$FP_{LRF} < C_0(x), C_1(x), C_{n\_r/2+1}(x), C_k(x r \overline{x})/C_k(x)/C_k(x) >$$

where

- n_r is the number row that correspond to the number of core-cells contained in a memory column;
- x, $\overline{x} \in \{0,1\}$ and $x \neq \overline{x}$;
- $n\_r/2 + 1 < k < n\_r$.

In FP_{LRF}, more than half of the core-cells of the memory column is initialized to logic value x. One of the remaining core-cell (C_k) of the column is initialized to logical value $\overline{x}$ and then the $r\overline{x}$ operation is performed in order to sensitize the faulty behavior. The value returned by the read is the logic value x instead of the expected $\overline{x}$.

The functional fault model of LRF is defined by means of the FP set obtained by considering the values associated to the variable x, n_r and k on FP_{LRF}. The importance of the masking effect leading to LRF is proportional to the number of core-cells storing a logic '0.' In order to identify the relation between the values stored in the core-cells of the column and the LRF occurrence, Dilillo et al. (2006c) propose SPICE simulations on a 65 nm 1024 × 1024 SRAM, with the following operation conditions:

- *Process corner:* typical
- *Supply voltage:* 0.7 V
- *Temperature:* 125°C
- *Cycle time:* 3 ns

The temperature is chosen at 125°C because a high temperature increases leakage currents, maximizing the occurrence of LFRs, as detailed in Section 7.2.2. The simulations study the variation of the differential value ΔBL between the two bit lines, useful for the read operation, for different distribution of '0' and '1' in the core-cells of the column. The results of these simulations are reported in Figs. 7.10 and 7.11.

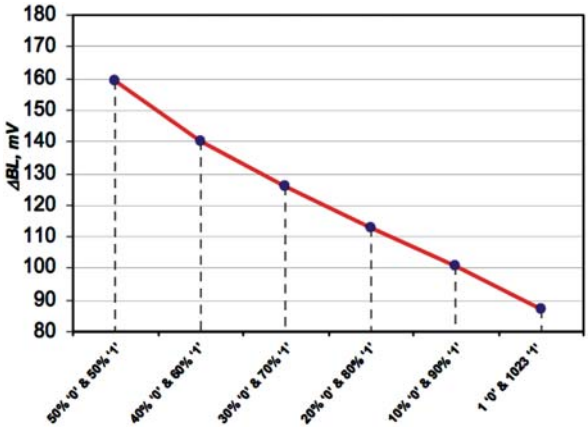


Fig. 7.10 ΔBL reduction during the read operation due to leakage currents, varying with the data distribution in memory column

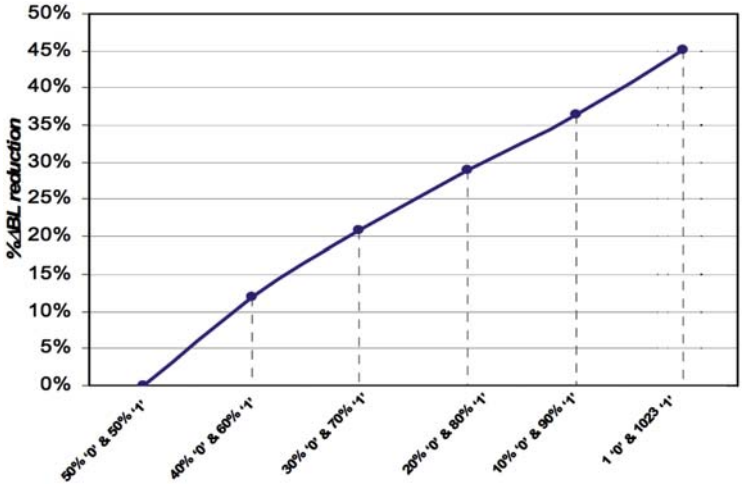


Fig. 7.11 %ΔBL reduction during the read operation due to leakage currents, varying with the data distribution in memory column

The graph in Fig. 7.10 shows the variation of the useful ΔBL in relation with the distribution of '0' and '1' stored in the column. The analysis of the graphs shows that the increase of the number of core-cells storing a logic '0' reduces linearly the value of the useful ΔBL. Two main scenarios can be identified:

- The *average case* is when in the column there are 50% of core-cells storing '0' and 50% of core-cells storing a logic '1.' In this configuration, the global leakage currents, through the core-cell pass-transistors, are symmetric, thus they have no effect on the read operation: the useful ΔBL is the highest (160 mV).
- The *worst case* is when 1,023 core-cells store a certain value, e.g., a logic '0,' and only one stores the opposite value, a logic '1.' During the reading of the core-cell storing a logic '1,' the leakage currents of all the unselected core-cells, storing a logic '0,' contribute to reduce the ΔBL useful for the read operation.

The graph in Fig. 7.11 shows the percentage of reduction of the ΔBL, useful for the read operation, in relation with the average case (50% of core-cells storing '0' and 50% of core-cells storing '1') for different distributions of stored values.

In the worst case, there is a reduction of ΔBL of about 45%, in relation with the average case, with a value of ΔBL very close to the safety value (VDD/10). The consequence is that, in the condition of reduced ΔBL, the memory is very sensitive to the noise and more prone to faults during the read operation. When ΔBL is close to the minimum safety value (VDD/10), any perturbation, such as VDD bounce, ground bounce, and VDD droop, may potentially lead to a faulty read, i.e., an output value different from the expected one may occur.

7.2.1 Analysis of Supply Voltage Variations

Let us now consider the impact of voltage supply on ΔBL (and so on the read operation) through the leakage current variation. Figures 7.12 and 7.13 show the results of SPICE simulations reported in Dilillo et al. (2006c) that refer to the 65 nm 1024 × 1024 SRAM introduced above, with the following operating conditions:

- *Process corner:* typical
- *Supply voltage:* 0.5–0.8 V
- *Temperature:* 125°C
- *Cycle time:* 3 ns

In Fig. 7.12, the curves at the top (a) and the bottom (b) refer to the equivalent overall leakage currents of 1,023 unselected core-cells (all in the same column) storing a logic '0' (the scale is on the right, A) through the pass-transistors on the nodes at logic '0,' and through the pass-transistors on the nodes at logic '1,' respectively. The curve marked with (c) shows the variation of ΔBL (the scale is on the left, mV) with supply voltage in the range from 0.5 to 0.8 V. The analysis of curve (a) demonstrates that the intensity of leakage currents on the node at logic '0' increases with the supply voltage, amplifying the masking effect on the read operation. Conversely, despite the higher leakage currents, the value of the useful ΔBL increases for higher supply voltages, as shown by curve (c). The worst case occurs for low values of VDD. In particular, for VDD = 0.5 V the value of ΔBL is clearly lower than VDD/10. This unexpected result is due to the fact that the increase of the read ΔBL is higher than the increase of leakage currents.

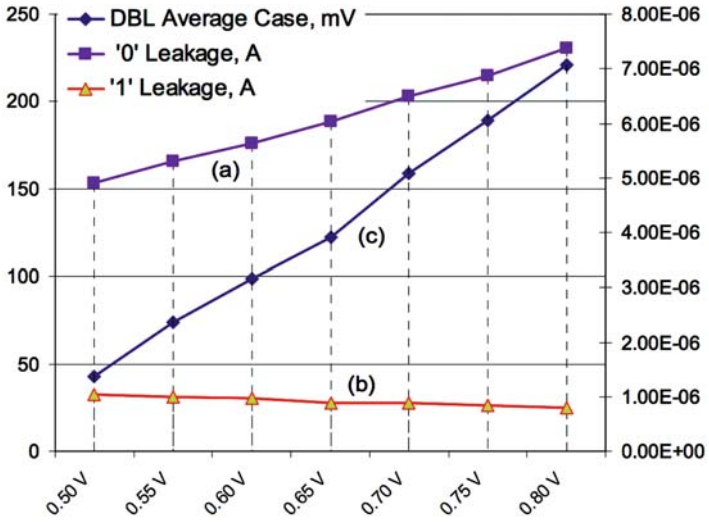


Fig. 7.12 Average Case read DBL(=ΔBL) and leakage currents through the pass-transistors of unselected core-cells in relation with VDD variation

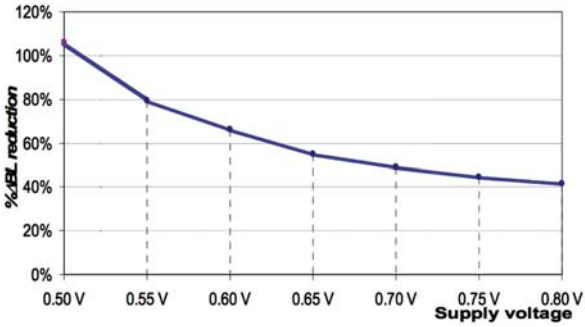


Fig. 7.13 %ΔBL reduction, due to leakage currents, in relation with the supply voltage

Thus, for higher supply voltage the incidence of the LRFs is lower. This is also confirmed by the graph in Fig. 7.13 that shows the reduction percentage of read ΔBL for different VDD levels. The curve shows how the reduction of ΔBL is more evident for lower supply voltage, regardless of the lower leakage currents.

7.2.2 Analysis of Temperature Variations

The variation of ΔBL with the temperature is now considered. The graphs in Figs. 7.14 and 7.15, reported in Dilillo et al. (2006c), show the results of SPICE simulations on the 65 nm 1024 × 1024 SRAM considered earlier, with the following operating conditions:

- *Process corner:* typical
- *Supply voltage:* 0.8 V
- *Temperature:* 25–125°C
- *Cycle time:* 3 ns

In Fig. 7.14, the two curves show the variation of ΔBL in the Average Case and Worst Case, with the temperature in the range from 25 to 125°C. The analysis of these two curves demonstrates that the read ΔBL decreases when the temperature increases. The influence of the leakage current in the reduction of the read ΔBL is more evident for higher temperature, because the distance between the two curves in the graph is larger for higher temperature. This result is more evident in the graph in Fig. 7.15, which shows the reduction percentage of read ΔBL with the temperature: the reduction is more than linear.

7.2.3 Test and Diagnosis of LRFs

Test algorithms that are able to cover LRFs, maximizing the fault incidence and coverage, must match the three following requirements (Dilillo et al. 2006c):

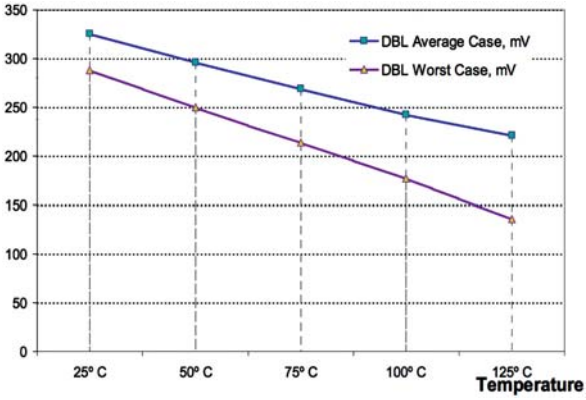


Fig. 7.14 Average Case read DBL(=ΔBL) and Worst Case DBL in relation with temperature

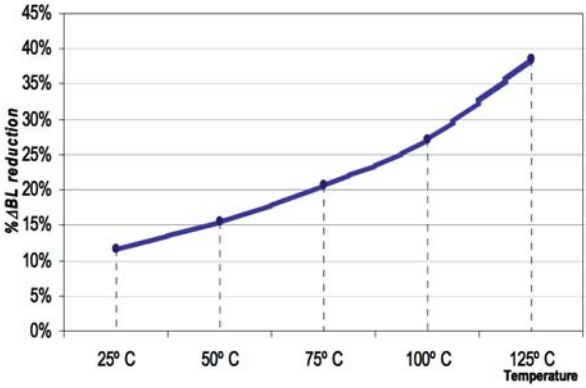


Fig. 7.15 %ΔBL reduction, due to leakage currents, in relation with the temperature

a. In the column composed of m core-cells, $m - 1$ core-cells have to store the same value $x \in \{0,1\}$ and one core-cell has to store $\overline{x}$, where $\overline{x}$ is the opposite of x;
b. When the requirement (a) is satisfied, a read operation is performed in the core-cell storing $\overline{x}$ ($= r\overline{x}$).
c. The requirement (a) and (b) have to be verified for all the columns in the memory array and for both logic values.

The requirement (a) is necessary for the sensitization of LRFs, while the second requirement (b) is necessary for both sensitization and observation. The third requirement (c) is necessary because in nanoscale technologies there are often fluctuations of different parameters (e.g., transistor threshold voltage V_t) even inside the die (Borkar et al. 2003). Thus, in the same memory array, there can be columns that present LRFs and fault-free columns. Consequently, the test needs to be run in all the columns of the memory array. The LRF is a dynamic fault since its sensitization needs more than one operation ($m + 1$). In the literature, there are many tests that meet the outlined three requirements. Among them, the most known March tests

are able to cover LRFs, but these tests generally target a large number of faults and consequently it can be difficult to use them for diagnostic purposes. For this reason, Dilillo et al. (2006c) proposes a March like test, named March LRF, which allows the test and diagnosis of LRFs. In order to make the various tests more effective in LRF coverage, it is possible to take into account the results on voltage and temperature analysis. In particular, it would be useful to run the tests in the conditions that maximize the presence of LRFs, i.e., low supply voltage and high temperature.

7.3 Complex Read Fault Analysis

All the memory elements that are involved in the read operation can be affected by electrical failure, resulting in Read Faults (RFs). Many studies target RFs due to a single electrical cause and the proposed detection algorithms are effective over a certain fault threshold. For example, certain algorithms cover RFs due to resistive-open defects when the equivalent resistance is above a certain value; other algorithms cover RFs due to variation of V_t when this parameter fluctuates over a certain percentage. Dilillo and Al-Hashimi (2007) shows that the collective action of different contributions to RFs may induce the read operation to fail, even when these contributions are individually not sufficient to generate malfunctions. These faults can be modeled as Complex Read Faults (CRFs).

The main elements that are responsible for the operation are the following ones:

- The core-cell, where the information is stored (Dilillo et al. 2005b).
- The word line that carries the signal enabling the core-cell selection for the read operation (Dilillo et al. 2005b).
- The bit lines that represent the wires that allow the exchange of the data to and from the core-cell through the core-cell pass-transistors (Mukhopadhyay et al. 2005, Agawa et al. 2001).
- The pre-charge circuit that sets the voltage level at VDD of the bit lines in order to ensure a correct read operation (Dilillo et al. 2005d).
- The sense amplifier that amplifies the read value from the bit lines to the output (Lovett et al. 2000, Chen et al. 2005).

These elements can be affected in different ways producing failures of the read operation that can be modeled as different RFs as RDF, DRDF, IRF, LRF, URRF, and URWF, which have been detailed previously in this book.

The various RFs may coexist and sum their effects. Most of test algorithms target only one RF model at a time and are effective only above a certain fault threshold. The simultaneous occurrence of multiple causes of RFs, below the detection threshold, may lead to read malfunctions, due to the cumulative effect. This means that in order to test efficiently the RFs it is useful to consider the combination of all the requirements to cover the different RFs. In nanoscale technologies, the occurrence of resistive-open defects is more and more frequent because of the reduction of the

wires dimension, their reduced thickness, and the presence of a high number of vias between the multiple levels of metal (Li et al. 2001). At the same time, leakage currents are a constant relevant factor for technology under 90 nm, with drawbacks in terms of power consumption (Shengqi et al. 2005, Sung et al. 2005) and reliability (Mukhopadhyay et al. 2005), imposing new solutions in terms of design (Agawa et al. 2001). These factors are the direct causes of the RF, and for their nature may coexist summing their effects, resulting in *Complex Read Faults*. If the presence of multiple resistive defects are unlikely to appear in a relative small region of the die, the effects on the read operation, due to a single defect in the core-core-cell (or pre-charge circuit or sense amplifier), are certainly to be summed to the effects of the leakage currents, always massively present in technologies under 90 nm. The same observation can be done for sense amplifier dissymmetry, caused by V_t variations and that can sum its effect on the read operation to those brought by leakage currents and/or resistive defects. The consequence is that in SRAM the read faults can hardly be associated to a single electric origin, but they need to be treated as Complex Read Fault. In order to generate an efficient test methodology to cover the Complex RFs, it is necessary to match all the requirements to cover all the read fault models at same time. The combination of all the various test requirements for each simple RF model can be summarized in following points (Dilillo and Al-Hashimi 2007):

a. At least a core-cell for each row of the memory array needs to be accessed for a write and read operation; the data stored is not relevant for the fault detection, i.e., the data to be written and read could be logic '0' or logic '1.'
b. In the column composed of m core-cells, m – 1 core-cells have to store the same value $x \in \{0,1\}$ and one core-cell has to store $\overline{x}$, where $\overline{x}$ is the opposite of x;
c. When the requirement (a) is satisfied, a read operation is performed in the core-cell storing $\overline{x}$ ($= r\overline{x}$).
d. The points (b) and (c) have to be verified for all the columns in the memory array and for both logic values.
e. In a memory column, a write operation wx is to be performed on a core-cell CCa immediately followed by read operation $r\overline{x}$ on another core-cell CCb of the same column, with $x \in \{0, 1\}$ and $\overline{x}$ is the opposite of x. This requirement has to be verified for all the columns in the memory array and for both logic values.

The first requirement is to cover the RFs connected to electrical malfunctions (presence of resistive-open defects) in word line and the core-cell. The following three requirements are needed to cover the RFs due to leakage currents from the bit lines through the core-cell pass-transistors. These requirements ensure the coverage of RFs due to malfunctions of the sense amplifier, while requirement (e) allows the coverage of RFs due to resistive defects in the pre-charge circuit.

On the basis of the requirements to cover CRFs, Dilillo and Al-Hashimi (2007) proposes a March like test, named March CRF, which allows the coverage of read faults in SRAMs, because it takes into account multiple electric causes. This test algorithm is depicted in Fig. 7.16.

Fig. 7.16 March CRF

$$\underset{M_0}{\Uparrow (w1)_i (w0)_{all\text{-}\{i,\, i\pm1\}}}; \underset{M_1}{\Uparrow (w0)_{i\pm1} (r1)_i};$$
$$\underset{M_2}{\Downarrow (w0)_i (w1)_{all\text{-}\{i,\, i\pm1\}}}; \underset{M_3}{\Downarrow (w1)_{i\pm1} (r0)_i}$$

The element M0 operates a $w1$ in the ith core-cell of each column and $w0$ in all the core-cells of the column, excluded the ith and the $(i \pm 1)$th ($i + 1$ for the columns with even address and $i - 1$ for the columns with odd address). The index $i \in \{0, 1, 2, \ldots, n\}$, where n is the number of columns in the memory array; i increases to one unit at the end of each March element process in a column operation. The element M1 operates a $w0$ in the $(i \pm 1)$th core-cell and $r1$ the ith core-cell. The following two elements do the same operations inverting the written and read data.

The complexity of the algorithm March CRF is very low: $2N + 2n$ ($\approx 2N$), where N is the number of core-cells and n is the number of columns in the memory. In order to make the March CRF more effective, it is useful to run the test in the conditions that maximize the effects of the resistive-open defects and leakage currents, i.e., low supply voltage, high temperature, and the highest operating frequency.

7.4 Conclusion

Technology scaling is responsible for several side effects on SRAM operation like V_t mismatches, leakage currents, and multiple-cause failures. V_t mismatches affect the SRAM core-cell, thus preventing a correct functioning during the write operation by provoking TFs and during the read operation by provoking dRDFs. Leakage current flowing through the pass gates of unselected core-cells influences the read operation causing LRFs. The leakage current effect added to low resistive-open defects and sense amplifier dissymmetry may lead to Complex Read Faults (CRFs), which contribute to reduce the success of the read operation. All these side effects have been studied in detail in this chapter, and corresponding test solutions have been described.

Chapter 8
Diagnosis and Design-for-Diagnosis

8.1 Diagnosis Methods

This section first presents basics on memory diagnosis based on signature analysis. It also shows how signatures can be extended to deal with dynamic fault models. Then, a diagnosis approach that represents an alternative to signature-based approaches is described. This diagnosis technique is based on the effect–cause paradigm already developed for logic design diagnosis.

8.1.1 Cause–Effect Approach: Signature-Based Diagnosis

8.1.1.1 Principle

Classical diagnostic techniques for memories are based on signatures analysis (Abramovici et al. 1990). 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.

The quality of a diagnosis is given by the diagnosability ratio, 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 diagnosability ratio 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.

For a given test algorithm, the fault dictionary can be generated by listing the fault models and their corresponding syndromes. To illustrate the signature-based diagnosis principle, let us consider the well-known March C-, whose structure is shown in Fig. 8.1.

$$\{\Updownarrow (w0) \Uparrow (r0,w1) \Uparrow (r1,w0) \Downarrow (r0,w1) \Downarrow (r1,w0) \Updownarrow (r0)\}$$
$$\text{M0} \quad \text{M1} \quad \text{M2} \quad \text{M3} \quad \text{M4} \quad \text{M5}$$

Fig. 8.1 March C- structure

A. Bosio et al., *Advanced Test Methods for SRAMs*,
DOI 10.1007/978-1-4419-0938-1_8, © Springer Science+Business Media, LLC 2010

The fault dictionary, limited to stuck-at (SAF) and transition (TF) faults, obtained with the March C-, is given in Table 8.1 (Li 2001b).

Table 8.1 Fault dictionary for the March C- algorithm

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

In Table 8.1, $R_i = 0$ (1) means that the ith read operation of the test algorithm has returned a correct (faulty) value for a specific core-cell. For example, the first read operation of the March C- (R_0 in Table 8.1) returns a faulty value only in case of a SAF1. Conversely, a stuck-at-0 fault (SAF0) corresponds to the failure of all $r1$ operations. The signature (March syndrome) of SAF0 is (01010). It is important to mention that the signature does not depend on the faulty core-cell (i.e., each faulty core-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 the existing studies on memory fault diagnosis target static faults such as stuck-at, transition, and coupling faults (Harutunyan et al. 2006, Appello et al. 2006, Al-Harbi et al. 2007). 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 diagnosability ratio.

8.1.1.2 Extension to Dynamic Fault Diagnosis

Dynamic faults have been considered for diagnosis purpose using fault dictionary only in Thakur et al. (2006) and Ney et al. (2008c). In Thakur et al. (2006), the authors target both static and dynamic faults and propose new algorithms: March SD and the *Aggressor Location Algorithm* (ALA). This approach can be considered as an ad hoc solution and can be classified among the techniques that introduce dedicated algorithms (and hence additional complexity) for diagnosis purpose.

In Ney et al. (2008c), a signature-based approach for both static and dynamic faults is proposed. It consists in the extension of the syndrome and is operated without modifying the test algorithm. For an easy understanding of this diagnosis approach, let us consider the dynamic fault called Un-Restored Write Fault (URWF) as case study. The URWF can be due to different electrical causes such as resistive-open defects in the write driver (*c.f.* Chapter 5) or in the pre-charge circuit (*c.f.* Chapter 3) of SRAMs. In the first case, the write driver is turned off too late (or even not turned off at all in case of a high resistive-open defects), inducing a wrong level on bit lines at the end of the pre-charge phase. In the second case, the pre-charge circuit itself is not strong enough to fully charge bit lines. The common effect in

both cases is that the final voltage level of the BLs is erroneous at the end of the pre-charge phase, and the following read operation fails.

An URWF requires a write operation on a core-cell CC_A immediately followed by a read operation with an opposite data in another core-cell CC_B. In case of an URWF due to a defect in a write driver, core-cells CC_A and CC_B must share the same I/O circuitry (i.e., the same write driver). Otherwise, when the defect inducing an URWF is located in a pre-charge circuit, CC_A and CC_B have to be in the same column.

By considering classical memory architectures, it is possible to determine the whole set of possible addressing situations:

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

By 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 I/O_i *w.r.t.* Ad_i.
- Ad_{i-1} belongs (or not) to the same column C_i *w.r.t.* Ad_i.
- Ad_{i+1} belongs (or not) to the same I/O circuitry I/O_i *w.r.t.* Ad_i.
- Ad_{i+1} belongs (or not) to the same column C_i *w.r.t.* Ad_i.

Combinations described above are summarized in Fig. 8.2. On the left side, the list of the different addressing configurations concerning the previous accessed core-cell Ad_{i-1} is presented. The next accessed core-cell Ad_{i+1} is considered on the right part of the scheme.

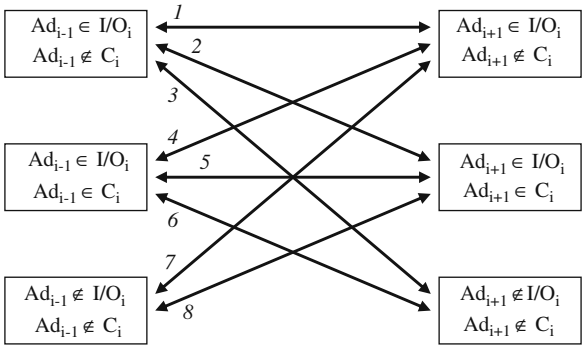


Fig. 8.2 Possible address sequence during test execution for URWF detection

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,

not only the previous accessed core-cell but also the next accessed core-cell have to be considered 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 Fig. 8.2. The knowledge of the addressing sequence allows determining the faulty memory element, i.e., pre-charge circuit or write driver. In accordance with the possible addressing configurations presented in Fig. 8.2, four cases are possible:

1. The configuration allows detecting URWFs caused by malfunction in a write driver.
2. The configuration allows detecting URWFs caused by malfunction in a pre-charge circuit.
3. The configuration allows detecting URWFs caused by malfunction of both pre-charge circuit and write driver but it cannot provide the exact failure localization.
4. The configuration allows detecting URWFs caused by malfunction in both pre-charge circuit and write driver but also provides the failure localization.

Table 8.2 List of extended signatures for URWF detection during March C- execution

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	00
	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 8.2 lists all possible extended signatures obtained with the application of the March C- algorithm and leading to URWFs detection. 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 Fig. 8.2 (labels on arrows). Columns from four to eight provide the classical March C- signatures for the URWF detection. Finally, the last two columns of Table 8.2 are fields 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 Fig. 8.2:

- 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, and '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, and 'x' if don't care).

For a better understanding of Table 8.2, let us consider the first line. The last two columns indicate that Ad_{i-1} and Ad_{i+1} belong to the same I/O than the current accessed core-cell, 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). Finally, the March C- application with this addressing configuration between Ad_{i-1}, Ad_i, and Ad_{i+1} provides the basic signature based on read operations (10100), listed in columns three to eight. Thus, the extended signature is '101001010.'

The list of extended signatures presented in Table 8.2 shows that:

- The addressing configurations 1, 3, and 7 allow the detection and the localization of an URWF in the write driver.
- The addressing configurations 5, 6, and 8 allow the detection of an URWF but do not provide any information on the failure localization. Hence 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 an 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.

For the diagnosis of other dynamic fault models, it will be 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 dynamic Read Destructive Faults (dRDF).

8.1.2 Effect–Cause Approach: History-Based Diagnosis

An alternative approach to achieve memory diagnosis can be borrowed from logic circuit diagnosis: the effect–cause approach (Abramovici et al. 1984). This approach analyzes the faulty responses of the test and uses the knowledge of the circuit in order to provide a set of suspected faults. The main advantage of this approach is that it is not limited to an a priori given group of fault models (i.e., a fault dictionary) on which the diagnosis is constructed. In practice, test responses (effect) are collected and then are analyzed in order to determine the source of the observed error (cause). This diagnosis method is called *history-based diagnosis* (Ney et al. 2008d).

8.1.2.1 Principle

The principle of the history-based diagnosis is based on the collection of two types of relevant information: (i) the faulty responses provided by the tester and (ii) 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 Fault Primitives (FPs) is generated.

As can be seen in Fig. 8.3, this diagnosis technique requires three inputs:

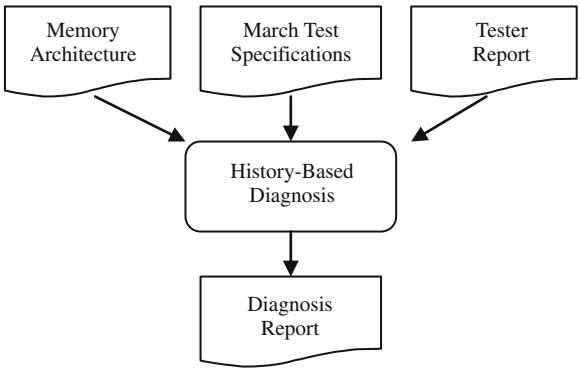


Fig. 8.3 History-based diagnosis principle

1. *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.
2. *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', etc.).
3. *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 reveals the fault and the corresponding core-cell address.

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

Step 1: History of Faulty Reads

This first step of the history-based 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,

all the operations previously performed on the affected core-cell until the last read operation that has returned a correct logic value are recorded. Let us illustrate this principle with a hypothetical 4 × 4 SRAM, whose scheme is presented in Fig. 8.4.

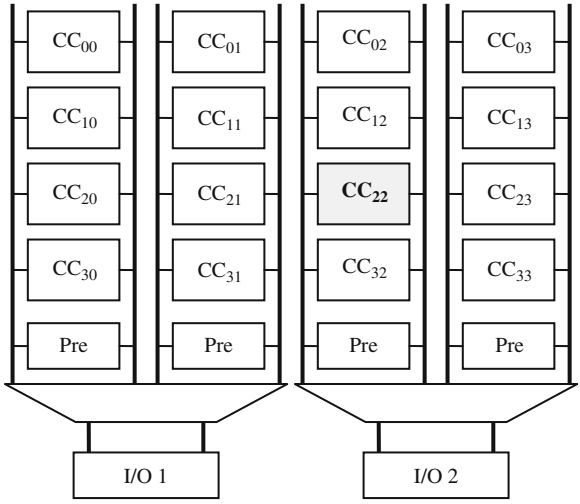


Fig. 8.4 A 4 × 4 memory core-cell array

Let us assume that this memory is affected by a TF0 (Transition Fault 1 to 0) in core-cell CC_{22}. The applied March algorithm is the March C- (see Fig. 8.1) 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, March test, and syndrome), the generation of the history record for each faulty read operation can start. The first faulty read operation is the $r0$ of the March element M3. The previous operations performed on CC_{22} were an $r1$ and a $w0$ of M2. The history record generation stops at the first previous read operation. The resulting history record of the first faulty read is denoted as H_0 and is composed as follows:

$$H_0 = 1w0r0$$

where '1' indicates the last value previously 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 all possible sequences of stimuli, denoted as S in Section 1.3.2, that can explain the faulty behavior on CC_{22} are computed and stored in a set denoted as eS_0 and composed as follows:

$$eS_0 = \{(1w0), (0r0)\}$$

In eS_0, '$1w_0$' can be interpreted as a failing $w0$ on CC_{22}. The second stimulus '$0r_0$' contains the actual faulty $r0$.

In the same way, H_1, related to the second faulty read on CC_{22} (with memory element M5), is computed and results in a new history record:

$$H_1 = 1w0r0$$

From H_1, eS_1 is computed as follows:

$$eS_1 = \{(1w0), (0r0)\}$$

Assuming that a memory component (core-cell, write driver, pre-charge circuit, etc.) may be affected by a single fault, the root cause of the observed error has to be present in all sets of stimuli. Consequently, the resulting eS_{faulty} related to CC_{22} is simply obtained by intersecting eS_0 and eS_1:

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

Step 2: History of Correct Read Operations

In order to reduce the set of stimuli included in eS_{faulty}, the sequence of stimuli that certainly cannot lead to a fault must be excluded. For this purpose, the sequence of stimuli generated by non-faulty read operations is considered. Considering again as example the TF0 on core-cell CC_{22}, one $r0$ operation among three is correct. Consequently, the history of this $r0$ is built as follows (in case of more than one correct $r0$, a history record should have been generated for each correct $r0$):

$$H_0' = xw0r0$$

with 'x' meaning that the content of the core-cell is unknown before starting the March element M0.

From H_0', $eS_{fault\text{-}free}$ is generated as follows:

$$eS_{fault\text{-}free} = \{(xw0), (0r0)\}$$

The stimuli included in $eS_{fault\text{-}free}$ are those for which no faulty behaviors have been observed. Note that the history of the two correct $r1$ is not considered because it could not reduce the number of stimuli.

Step 3: FP Compilation

After Steps 1 and 2, two sets of stimuli, eS_{faulty} and $eS_{fault\text{-}free}$, have been built. Now, sequences of stimuli composing $eS_{fault\text{-}free}$ have to be removed from eS_{faulty}. This

operation is reported in the following equation:

$$eS_{report} = eS_{faulty} - (eS_{faulty} \cap eS_{fault\text{-}free}) \quad (8.1)$$

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

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

This set of stimuli (in this case with only one element) represents the possible root causes of the observed errors.

From eS_{report}, the set of FPs including the set of stimuli must be computed. For each stimulus of eS_{report}, the corresponding FP is computed by considering the value observed during the read operation. The resulting FPs are stored in the set eFP_{report}.

From the above example, the obtained eFP_{report} is:

$$eFP_{report} = \{< 1w_0/1/- >\}$$

Note that since the sequence of operations does not contain any read operation, the value R is set to don't care '–' (*c.f.* Section 1.3.2).

Step 4: Fault Model Allocation

From eFP_{report}, the corresponding fault models can be associated to each FP. Let us consider again the above example, for which the reduced set of FPs is:

$$eFP_{report} = \{< 1w_0/1/- >\}$$

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 injected in the memory array on CC_{22}.

8.1.2.2 Extension to Dynamic Fault Diagnosis

As presented in Ney et al. (2008d), the history-based diagnosis approach, shown in the previous sub-section, is also able to diagnose dynamic faults if additional information are stored during the history computation. Hereafter, the extension of the history-based diagnosis approach for the following dynamic fault models is presented:

- Slow Write Driver Fault (SWDF).
- Un-Restored Write Fault (URWF).
- Un-Restored Destructive Write Fault (URDWF).
- dynamic Read Destructive Fault (dRDF).

As shown in Chapter 5, a SWDF 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:

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

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 the history-based diagnosis method consists in taking into account the operations performed on the faulty core-cell, additional information concerning the previous data written by the write driver must be also stored. Consequently, a first requirement that will be exploited by the history-based diagnosis method is defined as follows:

Requirement 1: During the diagnosis process, a record of the data previously written by the write driver must be done.

URWF and URDWFs are dynamic faults due to resistive-open defects in the pre-charge circuit (*c.f.* Chapter 3) and/or the write driver (*c.f.* Chapter 5). Both affect the read operation when it is preceded by a write operation. As presented in Chapter 5, the URWF and URDWF resulting from a resistive-open defect in the write driver require the following sequence of operations to be detected:

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

where both operations have to be performed on two distinct core-cells that belong to the same write driver.

As mentioned in Chapter 3, when an URWF is due to a resistive-open defect in the pre-charge circuit, the sensitization sequence is the same, but in this case, both operations have to be performed on two distinct core-cells belonging to the same column, i.e., to the same pre-charge circuit.

As for SWDF, a second requirement that will be exploited by the history-based diagnosis method is formulated as follows:

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

Finally, a dRDF is a dynamic fault affecting the core-cell. It has been shown in Chapter 2 that such fault model requires the following sensitizations sequence:

$$w\mathrm{x}\ r\mathrm{x}^{M}$$

Moreover, it has been shown in Chapter 2 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 the history-based diagnosis, the record of such stresses must be done in order to be able

to deal with dRDF diagnosis. As for the other dynamic fault models exposed above, a third requirement that will be exploited by the history-based diagnosis method is formulated as follows:

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

Based on requirements exposed above, the record of additional information during the diagnosis process must be done in order to cover dynamic faults. For this purpose, in the history record, an *Additional Information Vector* (AIV) is included, which stores information about the previously 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).
- One bit to report if Ad_{i-1} and Ad_i belongs to the same column (0: false, 1: true).
- The last operations ($r0$, $r1$, $w0$, or $w1$) performed on Ad_{i-1}.

Let us now illustrate the use of this AIV in two examples.

Example 1: dynamic Read Destructive Fault

Let us assume that the core-cell CC_{22} (see Fig. 8.4) is affected by a dRDF, i.e., one $r1$ just after a $w1$ causing the faulty swap of CC_{22} content. In this case, March C- is applied with a fast R addressing order as proposed in Dilillo et al. (2004b). 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:

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

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 an $r1$. All the other 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 the first term (RES^2) has been obtained. 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 eight 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 undergo any electrical stimulation during 16 clock cycles. At the end, CC_{22} is not accessed, even indirectly (RES), during 24 clock cycles implying B^{24} in H_0.

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, AIV_0 is computed as follows:

I/O	Column	Operation
0	0	$w0$

From H_0 and AIV_0, eS_0 is computed as follows:

$$eS_0 = \{(0w1), (1r1), (0w1r1), (0w1r1r1), (Ad_{i-1}(1w0), Ad_i(1r1))\}$$

The two first stimuli of eS_0 consider respectively the first and last operations of H_0 with '$0w1$' meaning that the $w1$ operation may have failed and '$1r1$' containing the actual faulty $r1$ operation.

Then eS_0 is completed with stimuli 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), the '$0w1r1$' and '$0w1r1r1$' FPs (considering RES $\approx$ read operation) are computed. Finally, the last FP of eFP_0 is obtained with the help of the AIV_0. The '$1w0$' is performed on a core-cell corresponding to the address Ad_{i-1}. The '$1r1$' is performed on the core-cell corresponding to the address Ad_i.

In the same way, H_1 related to the second faulty read is computed as follows:

$$H_1 = 0w1RES^4B^{24}RES^2r1$$

and AIV_1 related to H_1 is the following:

I/O	Column	Operation
1	0	$w0$

From H_1 and AIV_1, eS_1 is computed as follows:

$$\begin{aligned} eS_1 = \{&(0w1), (1r1), (0w1r1), (0w1r1r1) \\ &(0w1r1r1r1), (0w1r1r1r1r1), (Ad_{i-1}(1w0), Ad_i\,(1r1))\} \end{aligned}$$

Note that, Ad_{i-1} and Ad_i belong to the same I/O circuitry. The eS_{faulty} is obtained by intersecting eS_0 and eS_1:

$$eS_{faulty} = \{(0w1), (1r1), (0w1r1), (0w1r1r1)\}$$

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

$$eS_{report} = \{(0w1), (1r1), (0w1r1), (0w1r1r1)\}$$

From the eS_{report}, the set of FP is computed resulting in:

$$eFP_{report} = \{< 0w_1/1/- >, < 1r_1/0/0 >, < 1r_1/1/0 >, < 0w_1r_1/0/0 >, \\ < 0w_1r_1/1/0 >, < 0w_1r_1r_1/0/0 >, < 0w_1r_1r_1/1/0 >\}$$

Note that when the sequence of operation ends with a read operation, two FPs are generated. The first one considers that the value stored in the core-cell has been flipped due to the fault, while the second one only considers that the value returned by the read operation is faulty. For example, the sequence '$1r1$' corresponds to the two following FPs: $<1r_1/0/0>$ and $<1r_1/1/0>$.

The last step of the method assigns the fault models to the FPs found. From eFP_{report} fault candidates are TF1, SAF0, RDF (Read Destructive Fault), IRF (Incorrect Read Fault), dRDF (dynamic Read Destructive Fault), and dIRF (dynamic Incorrect Read Fault).

Example 2: Un-Restored Write Fault

As second example, let us consider the Un-Restored Write Fault (URWF). Such a fault model is caused by defects in the pre-charge circuit (*c.f.* Chapter 3) or in the write driver (*c.f.* Chapter 5). In this example, a defect is injected in the pre-charge circuit of the last column of the memory presented in Fig. 8.4. After applying the March C- with a fast C addressing order, the test report presents the following syndromes:

0 0 0 1 0 (Address CC03)
0 1 0 1 0 (Address CC13)
0 1 0 1 0 (Address CC23)
0 1 0 0 0 (Address CC33)

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:

$$H_0 = 0w1B^6RES^2B^6RES^2B^6RES^2B^6r1$$

and AIV_0 related to H_0 is the following:

I/O	Column	Operation
1	1	$w0$

From H_0 and AIV_0, eS_0 is computed:

$$eS_0 = \{(0w1), (1r1), (Ad_{i-1}(1w0), Ad_i(1r1))\}$$

with Ad_{i-1} and Ad_i belong to the same column. As there is only one faulty read, $eS_{faulty} = eS_0$. Now, the history of the remaining correct read is built as follows:

$$H_0' = 0w1B^6RES^2B^6RES^2B^6RES^2B^6r1$$

and AIV_0' related to H_0' is the following:

I/O	Column	Operation
1	0	$w0$

From H_0' and AIV_0', eS_0' is obtained as follows:

$$eS_0' = \{(0w1), (1r1), (Ad_{i-1}(1w0), Ad_i(1r1)\}$$

with Ad_{i-1} and Ad_i that belong to the same I/O circuitry. As there is only one correct read, $eS_{fault\text{-}free} = eS_0'$. From the intersection of eS_{faulty} and $eS_{fault\text{-}free}$, eS_{report} is obtained as follows:

$$eS_{report} = \{(Ad_{i-1}(1w0), Ad_i(1r1))\}$$

with Ad_{i-1} and Ad_i belong to the same column. From the eS_{report}, the set of FP is computed resulting in:

$$\begin{aligned} eFP_{report} = \{ &< Ad_{i-1}(1w0), Ad_i(1r1)/Ad_i(1)/Ad_i(0) > , \\ &< Ad_{i-1}(1w0), Ad_i(1r1)/Ad_i(0)/Ad_i(0) > \} \end{aligned}$$

The fault model related to the eFP_{report} can be either an URWF or URDWF. In addition, as the previous $w0$ operation has been performed on the same column, the observed error is due to a defect in the pre-charge circuit (un-correct bit line pull-up during the restoring phase between two operations). Consequently, as seen in Chapter 3, the fault model affecting the core-cell CC_{03} can only be a URWF.

The same procedure is followed for the syndromes of core-cells CC_{13} and CC_{23}:

$$eS_{report} = \{(0w1), (1r1), (Ad_{i-1}(1w0), Ad_i(1r1))\}$$

with Ad_{i-1} and Ad_i belongs to either the same column or same I/O. From the eS_{report}, the set of FP is computed resulting in:

$$\begin{aligned} eFP_{report} = \{ &< 0w_1/1/- > , < 1r_1/0/0 > , < 1r_1/1/0 > , \\ &< Ad_{i-1}(1w0), Ad_i(1r1)/Ad_i(1)/Ad_i(0) > , \\ &< Ad_{i-1}(1w0), Ad_i(1r1)/Ad_i(0)/Ad_i(0) > \} \end{aligned}$$

The fault models related to the eFP_{report} can be a TF1, SAF0, IRF, RDF, URWF, and URDWF.

Finally, for the last syndrome related to the core-cell CC_{33} is:

$$\begin{aligned} eFP_{report} = \{ &< Ad_{i-1}(1w0), Ad_i(1r1)/Ad_i(1)/Ad_i(0) >, \\ &< Ad_{i-1}(1w0), Ad_i(1r1)/Ad_i(0)/Ad_i(0) > \} \end{aligned}$$

with Ad_{i-1} and Ad_i that belong to either the same column or same I/O. The fault model related to the eFP_{report} can be either an URWF or an URDWF.

Table 8.3 reports the fault models indentified for each syndrome. An important result is that a unique fault model can explain all syndromes, the URWF related to a defect in the pre-charge circuit (denoted as $URWF_{PRE}$ in Table 8.3).

Table 8.3 List of fault models report for each syndrome

Core-cell	Fault model
CC_{03}	$URWF_{\mathbf{PRE}}$
CC_{13}	TF1, SAF0, IRF, RDF, $URWF_{\mathbf{PRE}}$, $URWF_{WD}$, URDWF
CC_{23}	TF1, SAF0, IRF, RDF, $URWF_{\mathbf{PRE}}$, $URWF_{WD}$, URDWF
CC_{33}	$URWF_{\mathbf{PRE}}$, $URWF_{WD}$, URDWF

8.1.2.3 Experimental Results

This sub-section first presents the effectiveness of the history-based diagnosis approach compared to the signature-based technique. Then, additional extensive results to prove the effectiveness of the history-based diagnosis solution are shown.

Table 8.4 presents the results given by both diagnosis solutions (signature-based and history-based). The first and second columns give the fault model injected (FMod) and its location (Location) in a hypothetical SRAM (a 512 × 512 SRAM

Table 8.4 Signature vs. history-based diagnosis

Fault model	Location	Test algorithm	Signature-based	History-based
SAF1	$CC_{99,3}$	March C-, fast C	SAF1	SAF1
			IRF	IRF
			RDF	RDF
TF0	$CC_{123,12}$	March C-, fast C	TF0	TF0
dRDF (3 *r*1)	$CC_{99,3}$	March C-, fast R	SAF0	SAF0
			TF1	TF1
			RDF	RDF
			IRF	IRF
				dRDF(1 *r*1, 6 *r*1)
URWF (*w*1, *r*0)	WD (C_{0-3})	March C-, fast C	SAF1	SAF1
			IRF	IRF
			RDF	RDF
				URWF (WD: *w*1, *r*0)

with 128 I/O blocks, i.e., one I/O for group of four columns). Column 3 specifies the applied test algorithm and the addressing order used during the test application (Test Algorithm). The last two columns give the diagnosis report with the two approaches.

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). For this first scenario, the two diagnosis solutions return the same list of fault candidates that contains the injected fault model (SAF1). Then, a TF0 has been injected in $CC_{123,12}$. For this 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 $r1$ following a $w1$. The employed test algorithm is again the March C-, with fast R (row after row) addressing order. As reported in the two last columns, the signature-based diagnosis reports four fault candidates (SAF0, TF1, RDF, and IRF), all static, and does not include the actually injected fault model. As suspects, the history-based solution returns not only static faults but also the dynamic fault dRDF with the reference (1 $r1$, 6 $r1$). The latter suggests that the content of $CC_{99,3}$ swaps due to a dRDF, after being accessed for 1–6 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. This fault is due to a resistive defect in the write driver (WD) used by the first four columns of the simulated SRAM ($C_{0–3}$). The applied algorithm is again March C- with fast C addressing order. Compared to the other scenarios, a syndrome is obtained for each core-cell connected to the defective write driver. The signature-based 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 history-based approach confirms the possible occurrence of SAF1, IRF, and RDF, suggested by the signature-based approach, but it also proposes the URWF as fault candidate. The URWF is the actual cause of all syndromes and is due to a defect in the write driver sensitized by the test pattern ($w1$, $r0$).

A larger quantity of experimental results that demonstrate the efficiency of the history-based diagnosis is now detailed. These results are the outcome of 1,000 fault-injection experiments. The faults are introduced in randomly chosen memory locations.

The results of the first set of experiments are summarized in Table 8.5. 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 history-based diagnosis. It also indicates if the actual injected fault is present in this set of suspects (Y: yes, N: no). The last column (L) gives the percentage of correct location

Table 8.5 Experimental results – March C-

Scenario	Location	Test	R	L
dRDF (1 *r*1)	Memory array	March C-, fast R	5.8 (Y)	89%
dRDF (1 *r*0)	Memory array	March C-, fast R	3.2 (Y)	75%
SWDF (*w*1, *w*0)	WD	March C-, fast R	2.25 (Y)	95%
SWDF (*w*0, *w*1)	WD	March C-, fast R	1.5 (Y)	95%
URWF (*w*1, *r*0)	WD	March C-, fast C	2, (Y)	97%
URWF (*w*0, *r*1)	WD	March C-, fast C	4.6 (Y)	30%
URWF (*w*1, *r*0)	PRE	March C-, fast C	2 (Y)	98%
URWF (*w*0, *r*1)	PRE	March C-, fast C	4.2 (Y)	58%
URDWF (*w*1, *r*0)	WD	March C-, fast C	2 (Y)	97%
URDWF (*w*0, *r*1)	WD	March C-, fast C	4.6 (Y)	30%

of the faulty memory component (the same faulty model can be related to different defective components).

A second set of experiments have been performed using a different March Test. In this case, the test algorithm is March AB-, shown in Fig. 8.5, that is a modified version of the March AB (Benso et al. 2005) able to cover the URDWF and URWF.

$$\underset{M0}{\{\Updownarrow(w0)} \underset{M1}{\Uparrow(r0,w1,r1,w1)} \underset{M2}{\Uparrow(r1,w0,r0,w0)} \underset{M3}{\Downarrow(r0,w1,r1,w1)} \underset{M4}{\Downarrow(r1,w0,r0,w0)} \underset{M5}{\Updownarrow(r0)\}}$$

Fig. 8.5 March AB-

Table 8.6 gives the results of these experiments exposed as in Table 8.4.

Table 8.6 Experimental results – March AB-

Scenario	Location	Test	R	L
dRDF (1 *r*1)	Memory array	March AB-, fast R	4.2 (Y)	92%
dRDF(1 *r*0)	Memory array	March AB-, fast R	2.2 (Y)	77%
SWDF (*w*1, *w*0)	WD	March AB-, fast R	1 (Y)	100%
SWDF (*w*0, *w*1)	WD	March AB-, fast R	1 (Y)	100%
URWF (*w*1, *r*0)	WD	March AB-, fast C	2 (Y)	99.2%
URWF (*w*0, *r*1)	WD	March AB-, fast C	2.8 (Y)	69.5%
URWF (*w*1, *r*0)	PRE	March AB-, fast C	2.8 (Y)	99.2%
URWF (*w*0, *r*1)	PRE	March AB-, fast C	2.75 (Y)	72%
URDWF (*w*1, *r*0)	WD	March AB-, fast C	2 (Y)	99.2%
URDWF (*w*0, *r*1)	WD	March AB-, fast C	2.8 (Y)	69.5%

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.

8.2 Design-for-Diagnosis of Write Drivers

The objective of this section is to present design-for-diagnosis techniques for write divers. Firstly, the conditions that have to be fulfilled to guarantee the fault-free functioning of a write driver are detailed. Then, two design-for-diagnosis techniques able to diagnose faulty but also weak write drivers are presented.

8.2.1 Requirements for Fault-Free Operations of a Write Driver

The fault-free operation of the SRAM write driver has already been described in Chapter 5. Based on this description, two important conditions that are needed to guarantee the fault-free behavior of the write driver can be enumerated – a logic and an analog conditions.

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, a first condition for a fault-free operation of the write driver can be extracted:

$$\mathrm{BL} \oplus \mathrm{BLB} = 1 \tag{8.2}$$

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.

Voltage levels on bit lines during write operations are a major concern when embedded memories are used for high safety applications (automotive, medical, etc.). In fact, over the lifetime of a product, memories are exposed to many phenomena (DC noise, coupling effects, etc.) 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 correct 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 BL and I_{idealH} for 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 viewed on waveforms in Fig. 8.6 where a $w0$ operation

is performed by a fault-free write driver (top of Fig. 8.6) and a weak write driver (bottom of Fig. 8.6).

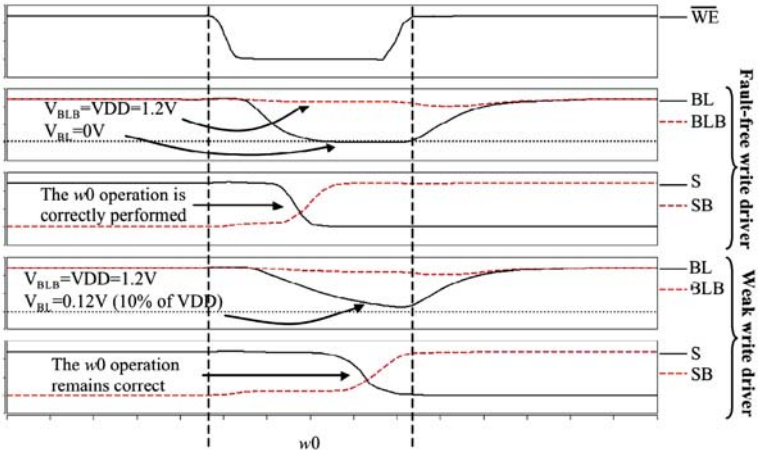


Fig. 8.6 Fault-free and weak write driver operations

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

$$I_{\text{realL}} \geq \alpha_1 \times I_{\text{idealL}} \text{ with } 0 \leq \alpha_1 \leq 1 \tag{8.3a}$$

$$\Rightarrow V_{\text{BL}} \leq \beta_1 \times \text{VDD with } 0 \leq \beta_1 \leq 1 \tag{8.3b}$$

and

$$I_{\text{realH}} \geq \alpha_2 \times I_{\text{idealH}} \text{ with } 0 \leq \alpha_2 \leq 1 \tag{8.4a}$$

$$\Rightarrow V_{\text{BLB}} \geq \beta_2 \times \text{VDD with } 0 \leq \beta_2 \leq 1 \tag{8.4b}$$

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 Equations (8.3a) and (<8.4a) with $\alpha_1 = 1$ and $\alpha_2 = 1$ implying $\beta_2 = 0$ ($V_{\text{BL}} = 0$ V) and $\beta_2 = 1$ ($V_{\text{BLB}} = \text{VDD}$).

Parameters α have to be selected depending on the memory technology and desired reliability level. In case of an SRAM designed with a 65-nm technology, these parameters have to be chosen as follows:

- α_1 insuring $V_{\text{BL}} \leq 0.1 \times \text{VDD}$
- α_2 insuring $V_{\text{BLB}} \geq 0.7 \times \text{VDD}$

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

8.2.2 Description of the Current-Based DfD Solution

The DfD solution proposed in Ney et al. (2008a) consists in adding a hardware module to verify both logic and analog conditions presented in the previous section. Note that, only the diagnosis of weak or wrong $w0$ operations is presented. The study of the $w1$ operation can be derived in a similar way.

The analog condition consists in verifying if the write driver delivers enough current in the bit lines. 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 reasons:

- The difficulty to design gates with very low or very high threshold voltages.
- The fact that two different voltages on each bit line to diagnose weak or wrong $w0$ and $w1$ operations must be sensed.

Consequently, in order to use simple CMOS gates to sense bit line voltage levels, the DfD solution presented in Ney et al. (2008a) proposes 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 Fig. 8.7.

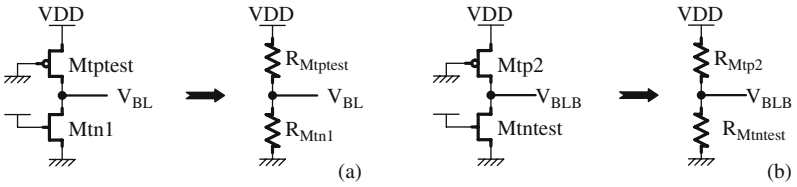


Fig. 8.7 Principle of the DFD solution (**a**) for the low level and (**b**) for the high level

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 $V_{BL} < \text{VDD}/2 \Rightarrow$ the write driver satisfies the analog condition.
- If $V_{BL} > \text{VDD}/2 \Rightarrow$ the write driver does not satisfy the analog condition.

To be more precise on the sizing of transistors Mtptest and Mtntest, let us consider Fig. 8.8. It represents I_{DS} as a function of V_{DS} voltage levels of Mtptest and Mtn1 transistors.

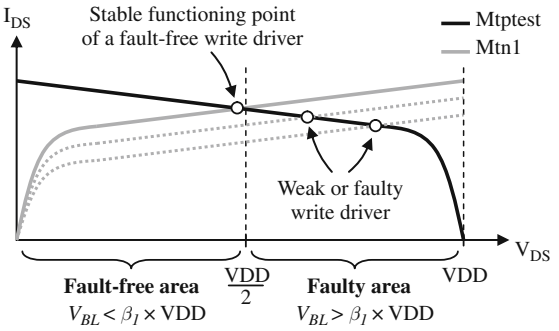


Fig. 8.8 Principle of the diagnosis solution

The hardware implementation of such a principle is presented in Fig. 8.9. 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 (/W0 and /W1 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. XOR and AND gates are used to provide the diagnosis results S. Node S must be at logic '1' during the write operation in case of a write driver satisfying the logic and analog conditions.

Waveforms in Fig. 8.10 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 /W0 = 0 and /W1=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.

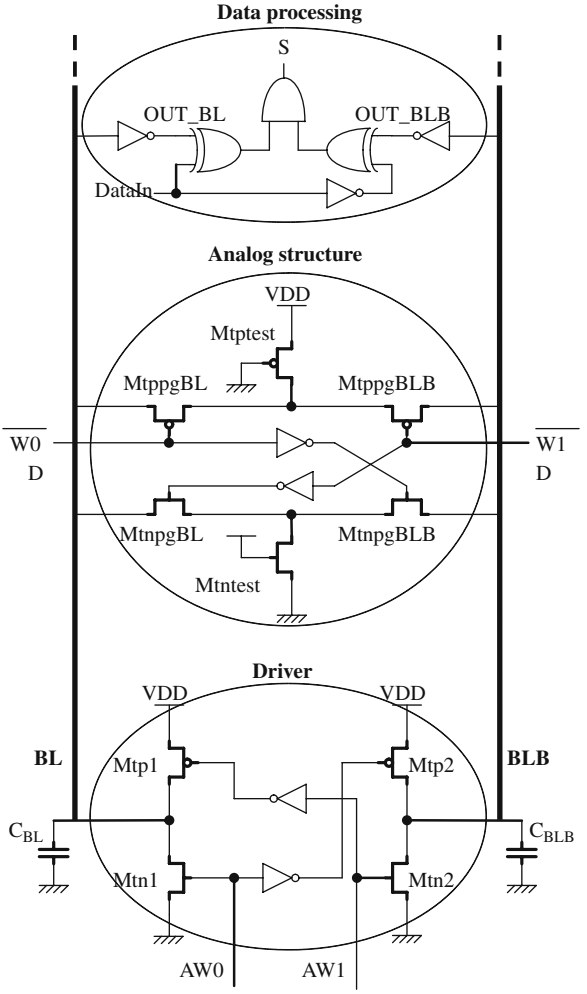


Fig. 8.9 Hardware implementation of the diagnosis module

8.2.3 Description of the Voltage-Based DfD Solution

Conversely to the current-based DfD solution, the voltage-based DfT solution proposed in Ney et al. (2008b) consists in controlling the actual voltage levels on BL and BLB during write operations. So, this DfD solution considers Equations (8.3b) and (8.4b) where β parameters, representing the level of charge and discharge, have to be chosen depending on the memory technology and the desired reliability level. As done for the current-based DfD solution, a 65-nm SRAM technology has been chosen with the following β parameters:

- $\beta_1 = 0.1$
- $\beta_2 = 0.2$

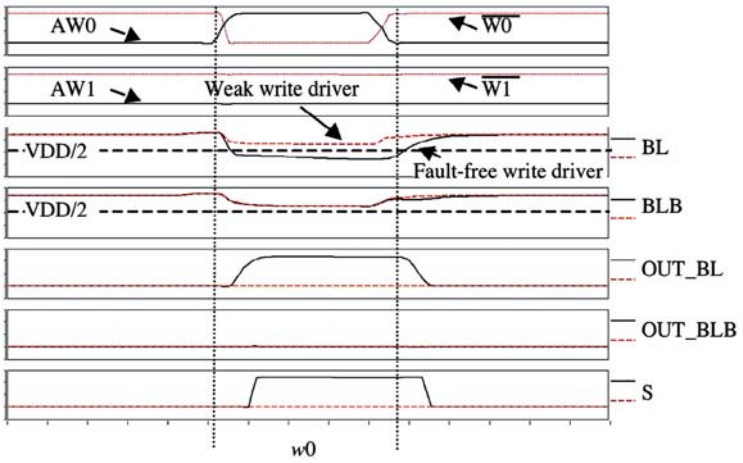


Fig. 8.10 Current-based DfD module functioning

Consequently, the write driver will be considered as faulty (i.e., 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.

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 and the other receiving BL or BLB levels. As shown in Ney et al. (2008b), a sense amplifier structure is the best candidate differential amplifier structures as its amplification is instantaneous and not linear as performed with operational amplifiers for example. Authors in Ney et al. (2008b) have also demonstrated that reusing the existing sense amplifier in the SRAM structure is unfeasible for the following reasons:

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

Consequently, they have proposed 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. This voltage-based DfD module uses two additional sense amplifiers connected to each write driver outputs (WD and WDB). Figure 8.11 shows the proposed 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 diagnosing $w0$ or $w1$ operations, respectively.

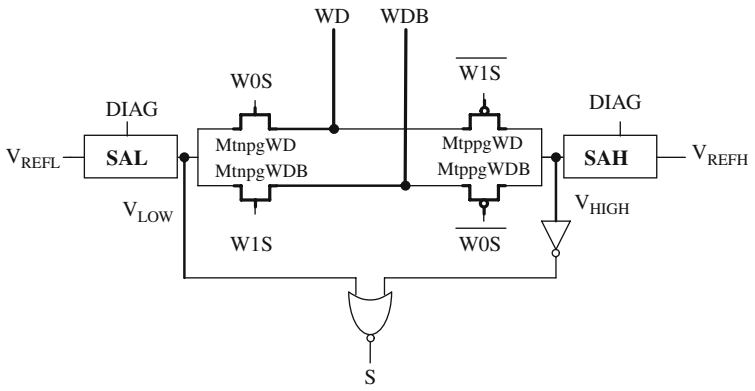


Fig. 8.11 Hardware implementation of the voltage-based DfD module

Each sense amplifier, SAL and SAH, receives the voltage level of the write driver (WD or WDB) on one input and a reference voltage ($V_{REFL} = 0.1 \times$ VDD and $V_{REFH} = 0.7 \times$ VDD) on the other. The DIAG signal allows activating both sense amplifiers during the diagnosis phase. Levels V_{LOW} and V_{HIGH} are the resulting amplification provided by each sense amplifier and represent the analog conditions.

The final diagnosis result (S) is a function (NOR gates) of both outputs V_{LOW} and V_{HIGH} in order to verify the logic condition. Table 8.7 provides the truth table of the DfD module. 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 8.7) 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.

Table 8.7 Truth table of the voltage-based DfD module

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

Waveforms in Fig. 8.12a, bshow the simulation results of the presented DfD module for a fault-free and a weak write driver, respectively.

On both simulations, a $w0$ operation is performed. Node WD is pulled down correctly in case of a fault-free write driver (Fig. 8.12a). 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 (Fig. 8.12a), $V_{LOW} < V_{REFL}$ and $V_{HIGH} > V_{REFH}$ thus implying $V_{LOW} = 0$ and $V_{HIGH} = 1$ when the DIAG signal is activated.

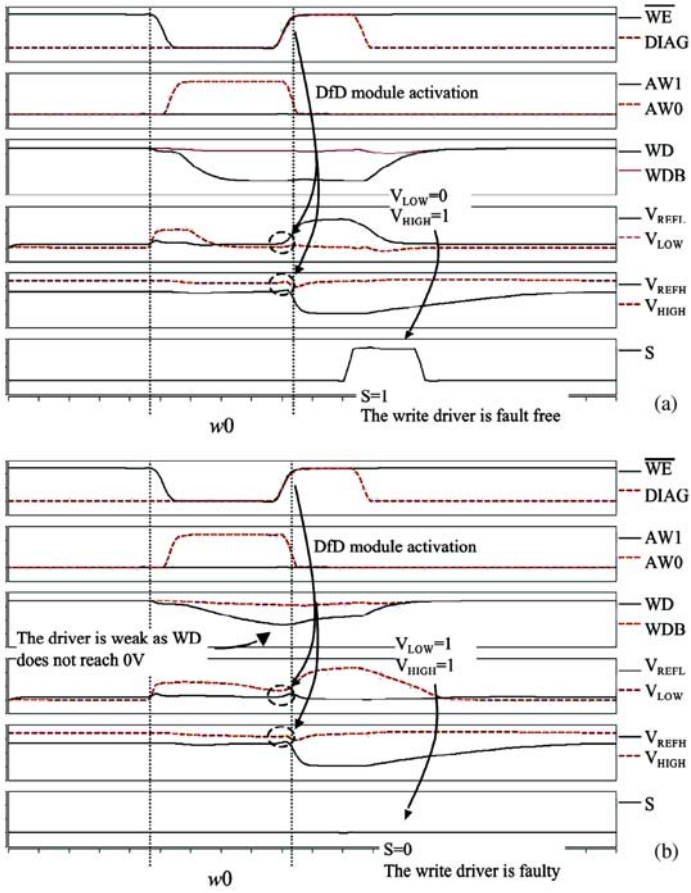


Fig. 8.12 DFD module functioning for (**a**) a fault-free and (**b**) a weak write driver

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 (Fig. 8.12b), $V_{LOW} > V_{REFL}$ and $V_{HIGH} > V_{REFH}$ thus implying $V_{LOW} = 1$ and $V_{HIGH} = 1$ when the DIAG signal is activated. Output *S* of the DfD module provides a logic '0' meaning that the write driver is faulty.

8.2.4 Diagnosis Sequence

Both presented DfD modules are able to verify the logic and analog conditions. Obviously, *w*0 and *w*1 operations are needed to diagnose the write driver. In this case, only defects involving a static behavior will be diagnosed. As shown in

Chapter 5, dynamic faults that may affect the write driver require two successive opposite write operations to be detected. Consequently, the diagnosis sequence able to deal with static and dynamic faulty behaviors as well is the following:

$$w\mathrm{x}\, w\bar{x}\, wx \tag{8.5}$$

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.

8.3 Conclusion

This chapter has first presented algorithmic diagnosis approaches for SRAMs. They are based on cause–effect (signature-based) and effect–cause (history-based) paradigms. The cause–effect approach uses a fault dictionary while the effect–cause approach 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. Comparative results shown in this chapter have demonstrated the interest of the history-based diagnosis solution to deal with dynamic faults. The second part of this chapter has been dedicated to design-for-diagnosis techniques for write drivers where two structures were detailed.

Summary

With the widespread and ever increasing use of semiconductor memories in electronic systems, memory test and diagnosis have always been a very hot topic in both industry and academia. As illustrated all along this book through the study of SRAMs, these devices are prone to defects since they are designed to exploit the technology limits and hence get the highest storage density and access speed.

The challenge of testing SRAMs consists in providing realistic fault models and test solutions with minimal test application time. However, the complexity of the memory devices makes fault modeling and testing not a trivial task. Classical memory test solutions cover the so-called static faults, but are not sufficient to cover dynamic faults that have emerged in the latest VDSM technologies.

This book presents a comprehensive state of the art of fault models and dedicated test and diagnostic solutions for the new class of dynamic faults occurring in the latest generation of SRAMs. After a first chapter giving the necessary background on SRAM fault modeling and testing, the book proposes a series of five chapters organized in a uniform manner. Each of them presents a complete study of dynamic faults in all memory components (core-cells, pre-charge circuits, address decoders, write drivers, sense amplifiers), starting from the electrical analysis of resistive defects, then describing the functional fault modeling process, and finally presenting the dedicated March test strategies. The two last chapters of the book are dedicated to an analysis of the impact of process variations and leakage current on the SRAM behavior and to an overview of diagnosis solutions for dynamic faults in SRAMs.

The authors believe that this book represents a useful material from different perspectives.

From a research point of view, the book can be used by research teams that work on the design, test, and reliability of memories. Its comprehensiveness can be useful to first understand the modus operandi that reveals how to model a memory fault starting from its electrical causes, and then reproduce the same process for further research activities.

From a teaching point of view, this book can be a manual for undergraduate and graduate students in microelectronics and electronic engineering, especially for supporting courses on electronic testing, microelectronics, computer architecture, and applied electronics. Furthermore, it can also be used for industrial training on

A. Bosio et al., *Advanced Test Methods for SRAMs*,
DOI 10.1007/978-1-4419-0938-1, © Springer Science+Business Media, LLC 2010

memory test. In this case, the book can be employed as a textbook for teaching the basics on memory architectures, to understand how an SRAM works at electrical and logical levels, or as a reference book on the test of SRAMs.

From an industrial point of view, this book will be of interest for the community of product managers, test and DfT engineers working in the memory market since quality and reliability of memories have important business implications. With the book, the reader can exploit all the presented test solutions for a direct implementation in an industrial product flow. The main benefits are test time reduction, minimized final product cost, and improved yield ramp-up.

As final remarks for this book, we briefly look at the various aspects about the future of memory testing. The challenges in this field will be driven by classical aspects of test development for this type of device: fault modeling, test algorithm generation, BIST for practical and cost-efficient at-speed testing, and BISR for improvement of yield and system reliability (Hamdioui et al. 2004). In addition to these classical aspects, test automation and emerging memory technologies will also represent challenging tasks. They are discussed below.

With the advancing technology scaling, the sensitivity of memories to physical defects and process variations will continue to grow, thus leading to new faulty behaviors that will require new test solutions. This will enforce the need of defect-based test approaches as those described in this book, as opposed to fault-based test approaches where only functional fault models are used to define functional test algorithms independent of the target technology. We believe that the need to work with electrical-level memory models and realistic fault models for specific memory architectures and technologies will be even more crucial to cover the wider and wider spectrum of failure mechanisms of future memories. In the meantime, defect-based memory testing will also need a strong investment in automating all steps of the testing process, from defect analysis and simulation, to realistic fault modeling and test generation (Di Carlo and Prinetto 2009). This automation process to generate efficient defect-based test approaches capable of dealing with the huge complexity of future memories will be a mandatory step for a complete industrialization.

Another important aspect of the future of memory testing is related to future technologies that will be used to satisfy the request for huge memory storage capacity and performance of the future electronic world featured by the "portable and converging" trend. After SRAM, DRAM, all types of ROM, and Flash memories, new technologies have emerged in the last years, mainly consisting in the four following families (Ginez et al. 2009): *Magnetic Random Access Memories* (MRAMs), *Ferroelectric Random Access Memories* (FeRAMs or FRAMs), *Phase-Change Random Access Memories* (PCMs or PRAMs), and *Resistive switching Random Access Memories* (ReRAMs). Although MRAMs and ReRAMs are still promising technologies, PRAMs and FRAMs are non-volatile memories which are very close to mass production, although improvements in the structure and the manufacturing process are still needed (Kim and Lee 2007). For all these devices, failure analysis, fault modeling, test, and diagnosis solutions (for repair or online

reconfiguration) will be needed, thus appealing for new research activities in the field.

Memory testing is an active area of research and development that has steadily moved from research labs to practice in the past decades. This book has detailed the advanced techniques to deal with dynamic faults. With the ongoing advances in technology, more innovation for memory testing will happen; in this respect, we hope that this book will serve as an inspiration for future research and development in the field.

References

Abramovici M, Menon PR, Miller DT (1984) Critical path tracing – an alternative to fault simulation. IEEE Des Test Comput 1(1):83–92

Abramovich M, Breuer MA, Friedman AD (1990) Digital system testing and testable design. IEEE Press, Piscataway, NJ. ISBN: 0-7803-1062-4

Adams RD (2002) High performance memory testing. Kluwer Academic Publishers, Hingham, MA. ISBN 1-4020-7255-4

Adams RD, Cooley ES (1996) Analysis of deceptive destructive read memory fault model and recommended testing, In Proceedings of IEEE North Atlantic Test Workshop, pp 27–32

Adams RD, Cooley ES (1997) False write through and un-restored write electrical level fault models for SRAMs. In Proceedings of IEEE International Workshop on Memory Technology, Design and Testing, pp 27–32

Agawa K, Hara H, Takayanagi T, Kuroda T (2001) A bitline leakage compensation scheme for low-voltage SRAMs. IEEE J Solid-State Circ 36(5):726–734

Al-Ars Z, van de Goor AJ (2001) Static and dynamic behavior of memory cell array opens and shorts in embedded DRAMs. In Proceedings of EDAA Design Automation and Test in Europe, pp 496–503

Al-Harbi SM, Gupta SK (2001) An efficient methodology for generating optimal and uniform March tests. In Proceedings of IEEE VLSI Test Symposium, pp 231–237

Al-Harbi SM, Noor F, Al-Turjman FM (2007) March DSS: a new diagnostic March test for all memory simple static faults. IEEE Trans Comput Aided Des Integr Circ Syst 26(9): 1713–1720

Appello D, Tancorre V, Bernardi P, Grosso M, Rebaudengo M, Sonza Reorda M (2006) Embedded memory diagnosis: an industrial workflow. In Proceedings of IEEE International Test Conference, pp 1–9

Bastian M, Gouin V, Girard P, Landrault C, Ney A, Pravossoudovitch S, Virazel A (2007) Influence of threshold voltage deviation on 90 nm SRAM core-cell behavior. In Proceedings of IEEE Asian Test Symposium, pp 501–504

Benso A, Di Carlo S, Di Natale G, Prinetto P (2002) Specification and design of a new memory fault simulator. In Proceedings of IEEE Asian Test Symposium, pp 92–97

Benso A, Bosio A, Di Carlo S, Di Natale G, Prinetto P (2005) March AB, March AB1: new March tests for unlinked dynamic memory faults. In Proceedings of IEEE International Test Conference, pp 1–6

Benso A, Bosio A, Di Carlo S, Di Natale G, Prinetto P (2008) March test generation revealed. IEEE Trans Comput 57(12):1704–1713

Bhavnagarwala AJ, Tang X, Meindl JD (2001) The impact of intrinsic device fluctuations on CMOS SRAM cell stability. IEEE J Solid-State Circ 36(4):658–665

Borkar S, Karnik T, Narendra S, Tschanz J, Keshavarzi A, De V (2003) Parameter variations and impact on circuits and microarchitecture. In Proceedings of EDAA Design Automation Conference, pp 338–342

Borri S, Hage-Hassan M, Dilillo L, Girard P, Pravossoudovitch S, Virazel A (2005) Dynamic fault models for embedded-SRAMs: analysis and test. J Electron Test: Theory Appl Springer, 21:169–178

Chen Q, Mahmoodi H, Bhunia S, Roy K (2005) Modeling and testing of SRAM for new failure mechanisms due to process variations in nanoscale CMOS. In Proceedings of IEEE VLSI Test Symposium, pp 292–297

Cheng K-L, Wang C-W, Lee J-N, Chou Y-F, Huang C-T, Wu C-W (2002) Fault simulation and test algorithm generation for random access memories. IEEE Trans Comput Aided Des Integr Circ Syst 21(4):480–490

Di Carlo S, Prinetto P (2009) Models in memory testing. In: HJ Wunderlich (ed) Models in hardware testing. Springer, Berlin

Dilillo L, Al-Hashimi B (2007) March CRF: an efficient test for complex read faults in SRAM memories. In Proceedings of IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, pp 173–178

Dilillo L, Girard P, Pravossoudovitch S, Virazel A, Borri S (2003) Comparison of open and resistive-open defect test conditions in SRAM address decoders. **In Proceedings of IEEE Asian Test Symposium, pp 250–255

Dilillo L, Girard P, Pravossoudovitch S, Virazel A, Hage-Hassan M (2004) Resistive-open defects in embedded-SRAM core cells: analysis and March test solution. In Proceedings of IEEE Asian Test Symposium, pp 166–271

Dilillo L, Girard P, Pravossoudovitch S, Virazel A, Borri S (2004a) March iC-: an improved version of March C- for ADOFs detection. In Proceedings of IEEE VLSI Test Symposium, pp 129–134

Dilillo L, Girard P, Pravossoudovitch S, Virazel A, Borri S, Hage-Hassan M (2004b) Dynamic read destructive fault in embedded-SRAMs: analysis and March test solution. In Proceedings of IEEE European Test Symposium, pp 140–145

Dilillo L, Girard P, Pravossoudovitch S, Virazel A, Hage-Hassan M (2005a) Data retention fault in SRAM memories: analysis and detection procedures. In Proceedings of IEEE VLSI Test Symposium, pp 183–188

Dilillo L, Girard P, Pravossoudovitch S, Virazel A, Borri S, Hage-Hassan M (2005b) Efficient March test procedure for dynamic read destructive fault detection in SRAM memories. J Electron Test: Theory Appl Springer, 21(5):551–561

Dilillo L, Girard P, Pravossoudovitch S, Virazel A, Hage-Hassan M (2005c) Resistive-open defect injection in SRAM core-cell: analysis and comparison between 0.13 μm and 90 nm technologies. In Proceedings of ACM IEEE Design Automation Conference, pp 857–862

Dilillo L, Girard P, Pravossoudovitch S, Virazel A, Bastian M (2005d) Resistive-open defect influence in SRAM pre-charge circuit: characterization and analysis. In Proceedings of IEEE European Test Symposium, pp 116–121

Dilillo L, Girard P, Pravossoudovitch S, Virazel A, Bastian M (2006a) March pre: an efficient test for resistive-open defects in the SRAM pre-charge circuit. In Proceedings of IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, pp 254–259

Dilillo L, Girard P, Pravossoudovitch S, Virazel A, Borri S, Hage Hassan M (2006b) ADOFs and resistive-ADOFs in SRAM address decoders: test conditions and March solutions. J Electron Test: Theory Appl Springer, 22(3):pp 287–296

Dilillo L, Al-Hashimi B, Rosinger P, Girard P (2006c) Leakage read fault in nanoscale SRAM: analysis, test and diagnosis. In Proceedings of IEEE International Design and Test Workshop

Dilillo L, Girard P, Pravossoudovitch S, Virazel A, Borri S, Hage-Hassan M (2007) Analysis and test of resistive-open defects in SRAM pre-charge circuits. J Electron Test: Theory Appl Springer, 23(5):435–444

Dilillo L, Girard P, Landrault C, Pravossoudovitch S, Virazel A, Bastian M, Gouin V (2008) Impact of technology scaling on defects and parameter deviations in embedded SRAMs. In Proceedings of IEEE VLSI Test Symposium, pp 336–336

Ding L, Mazumder P (2003) The impact of bit-line coupling and ground bounce on CMOS SRAM performance. In Proceedings of IEEE International Conference on VLSI Design, pp 234–239

Ginez O, Portal JM, Muller C (2009) Design and test challenges in resistive switching RAM (ReRAM): an electrical model for defect injections. In Proceedings of IEEE European Test Symposium, pp 61–66

Hamdioui S (2004) Testing static random access memories. Kluwer Academic Publishers, Hingham, MA. ISBN: 1-4020-7752-1

Hamdioui S, van de Goor AJ (2000) An experimental analysis of spot defects in SRAMs: realistic fault models and tests. In Proceedings of IEEE Asian Test Symposium, pp 131–138

Hamdioui S, Al-Ars Z, van de Goor AJ (2002) Testing static and dynamic faults in random access memories. In Proceedings of IEEE VLSI Test Symposium, pp 395–400

Hamdioui S, Gaydadjiev GN, van de Goor AJ (2004) The state-of-the-art and future trends in testing embedded memories. In Proceedings of IEEE International Workshop on Memory Technology, Design, and Testing, pp 54–59

Harutunyan G, Vardanian VA, Zorian Y (2006) Minimal March-based fault location algorithm with partial diagnosis for all static faults in random access memories. In Proceedings of IEEE Design and Diagnostics of Electronic Circuits and Systems, pp 260–265

Harutunyan G, Vardanian VA, Zorian Y (2007) Minimal March tests for detection of dynamic faults in random access memories. J Electron Test: Theory Appl 23(1):55–74

International Technology Roadmap for Semiconductors (ITRS) (2007) edition

Li JCM, Tseng C-W, McCluskey EJ (2001a) Testing for resistive opens and stuck opens. In Proceedings of IEEE International Test Conference, pp 1049–1058

Li JF, Cheng K-L, Huang C-T, Wu C-W (2001b) March-based RAM diagnosis algorithms for stuck-at and coupling faults. In Proceedings of IEEE International Test Conference, pp 758–767

Klaus M, van de Goor AJ (2001) Test for resistive and capacitive defects in address decoders. In Proceedings of IEEE Asian Test Symposium, pp 31–36

Lovett SJ, Gibbs GA, Pancholy A (2000) Yield and matching implications for static RAM memory array sense-amplifier design. J Solid-State Circ 35(8):1200–1204

Marinescu M (1982) Simple and efficient algorithms for functional RAM testing. In Proceedings of IEEE International Test Conference, pp 236–239

Mukhopadhyay S, Mahmoodi H, Roy K (2005) Modeling of failure probability and statistical design of SRAM array for yeld enhancement in nanoscaled CMOS. IEEE Trans CAD 24(12):1859–1880, December

Ney A, Girard P, Pravossoudovitch S, Virazel A, Hage-Hassan M (2007) Un-restored destructive write faults due to resistive-open defects in the write driver of SRAMs. In Proceedings of IEEE VLSI Test Symposium, pp 361–368

Ney A, Girard P, Landrault C, Pravossoudovitch S, Virazel A, Bastian M (2007a) Slow write driver faults in 65 nm SRAM technology: analysis and March test solution. In Proceedings of EDAA Design Automation and Test in Europe, pp 528–533

Ney A, Girard P, Landrault C, Pravossoudovitch S, Virazel A, Bastian M (2007b) Dynamic two-cell incorrect read fault due to resistive-open defects in the sense amplifiers of SRAMs. In Proceedings of IEEE European Test Symposium, pp 97–102

Ney A, Girard P, Landrault C, Pravossoudovitch S, Virazel A, Bastian M, Gouin V (2008a) A design-for-diagnosis technique for SRAM write drivers. In Proceedings of EDAA Design Automation and Test in Europe, pp 1480–1485

Ney A, Girard P, Landrault C, Pravossoudovitch S, Virazel A, Bastian M, Gouin V (2008b) An SRAM design-for-diagnosis solution based on write driver voltage sensing. In Proceedings of IEEE VLSI Test Symposium, pp 89–94

Ney A, Bosio A, Dilillo L, Girard P, Pravossoudovitch S, Virazel A (2008c) A signature-based approach for diagnosis of dynamic faults in SRAMs. In Proceedings of IEEE International Conference on Design and Technology of Integrated Systems, pp 1–6

Ney A, Bosio A, Dilillo L, Girard P, Pravossoudovitch S, Virazel A, Bastian M (2008d) A history-based diagnosis technique for static and dynamic faults in SRAMs. In Proceedings of IEEE International Test Conference, paper 3.2

Ney A, Girard P, Landrault C, Pravossoudovitch S, Virazel A, Bastian M (2009) Analysis of resistive-open defects in SRAM sense amplifiers. IEEE Trans VLSI Syst. DOI: 10.1109/TVLSI.2008.2005194 to appear in.

Nicolaidis M (1985) An efficient built-in self-test scheme for functional test of embedded memories. In Proceedings of International Symposium Fault Tolerant Computing, pp 118–123

Niggemeyer D, Rudnick EM (2004) Automatic generation of diagnostic memory tests based on fault decomposition and output tracing. IEEE Trans Comput 53(9):1134–1146

Niggemeyer D, Redeker M, Otterstedt J (1998) Integration of non-classical faults in standard March tests. In Proceedings of IEEE International Workshop on Memory Technology Design and Testing, pp 91–96

Otterstedt J, Niggemeyer D, Williams TW (1998) Detection of CMOS address decoder open faults with March and pseudo random memory tests. In Proceedings of IEEE Int. Test Conference, pp 53–62

Prince B (1996) Semiconductor memories: a handbook of design, manufacture and application, 2nd edn. Wiley, Malden. ISBN: 978-0-471-94295-5

Prince B (2002) Emerging memories: technologies and trends. Springer, Berlin. ISBN 0-7923-7684-6

Riedel M, Rajski J (1995) Fault coverage analysis of RAM test algorithms. In Proceedings of IEEE VLSI Test Symposium, pp 227–234

Rodriguez R, Volf P, de Gyvez JP (2002) Resistance characterization for weak open defects. IEEE J Des Test Comput 19(5):18–26

Sachdev M (1996) Test and testability techniques for open defects in RAM address decoders. In Proceedings of IEEE European Design and Test Conference, pp 428–434

Sachdev M (1997) Open defects in CMOS RAM address decoders. IEEE J Des Test Comput 14(2):26–33

Sedra AS, Smith KC (1998) Microelectronic circuit, 4th edn. Oxford University press, Oxford, UK. ISBN: 0-1951-4251-9

Senthinathan R, Prince JL (1991) Simultaneous switching ground bounce noise calculation for packaged CMOS devices. IEEE J Solid-State Circ 26(11):1724–1728

Sharma AK (1997) Semiconductor memories: technology, testing and reliability. IEEE Press, Piscataway, NJ. ISBN 0-7803-1000-4

Shengqi Y, Wolf W, Vijaykrishnan N, Yuan X, Wenping W (2005) Accurate stacking effect macro-modeling of leakage power in sub-100 nm circuits. In Proceedings of International Conference on VLSI Design, pp 165–170

Sung KN, Blaauw D, Mudge T (2005) Quantitative analysis and optimization techniques for on-chip cache leakage power. IEEE Trans VLSI Syst 13(10):1147–1156

Taur Y, Ning T (1998) Fundamentals of modern VLSI devices. Cambridge University Press, Cambridge. ISBN: 0-5215-5959-6

Thakur SK, Parekhji R, Chandorkar AN (2006) On-chip test and repair of memories for static and dynamic faults. In Proceedings of IEEE International Test Conference, pp 1–10

van de Goor AJ (1998) Testing semiconductor memories: theory and practice. COMTEX Publishing, Gouda, The Netherlands. ISBN 0-471-92586-1

van de Goor AJ, Al-Ars Z (2000) Functional memory faults: a formal notation and a taxonomy. In Proceedings of IEEE VLSI Test Symposium, pp 281–289

van de Goor AJ, Smit B (1994) The automatic generation of March tests. In Proceedings of IEEE International Workshop on Memory Technology Design and Testing, pp 86–91

van de Goor AJ, Tlili IBS (2003) A systematic method for modifying March tests for bit-oriented memories into tests for word-oriented memories. IEEE Trans Comput 52(10):1320–1331

van de Goor AJ, Hamdioui S, Wadsworth R (2004) Detecting faults in the peripheral circuits and an evaluation of SRAM tests. In Proceedings of IEEE International Test Conference, pp 114–123

Wang B, Yang J, Ivanov A (2003) Reducing test time of embedded SRAMs. In Proceedings of IEEE International Workshop on Memory Technology, Design and Testing, pp 47–52

Wu C-F, Huang C-T, Wu C-W (1999) RAMSES: a fast memory fault simulator. In Proceedings of IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp 165–173
Zarrineh K, Upadhyaya SJ, Chakravarty S (1998) A new framework for generating optimal march tests for memory arrays. In Proceedings of IEEE International Test Conference, pp 73–82
Zarrineh K, Deo AP, Adams RD (2000) Defect analysis and realistic fault model extensions for static random access memories. In Proceedings of IEEE International. Workshop on Memory, Technology, Design and Testing, pp 119–124

Index